情報技術者試験受験テキスト

開発技術

令和元年８月発行

KMC学習所

txt0400 開発技術目次

txt0400　開発技術

txt04011　開発企画

txt040111 システム企画・開発

①　システムエンジニアリング

ⓐ　コンピュータシステムとは

コンピュータシステムは、ソフトウェア、ハードウェア、ネットワーク、データベース、ドキュメント、手順、ユーザやオペレータなどのシステム要素を利用して情報を処理し、決められた目標を達成するための仕組である。

銀行のＡＴＭシステムは、ユーザの「通帳記入」要求に対して、端末やネットワーク、ソフトウェア、ハードウェア、データベースなどの各種要素を使用して、情報変換し、処理結果を返信し、通帳に記入する。

ⓑ　システムエンジニアリング

システムエンジニアリングは、さまざまな要素に注目し、それらの分析、設計を行い、製品やサービス、情報を変換・制御するシステムとして設計するプロセスである。

システムエンジニアリングのプロセスは、開発作業の対象が企業の場合、ビジネスプロセスエンジニアリングとよび、携帯電話やロボットなどの製品を対象にする場合、プロダクトエンジニアリングと呼ぶ。どちらのプロセスも特定の対象に合わせて、ソフトウェアの役割を定め、ソフトウェアとシステムの他の要素とを結びつけるための作業を行うものである。

ⓒ　システム開発作業の３段階

㋐　プロセス

プロセスはシステム開発作業を役割の観点からまとめたもので、互いに関連をもったアクティビティの集合であり、入力を出力に変換するものである。企画、要件定義、開発、運用、保守などの入力から出力へ変換させるもの、構成管理や文書作成、契約変更管理などの状態の変化に対応するもの、レビュー、検証、監査などの生産物や作業の評価などが含まれる。

㋑　アクティビティ

アクティビティは相関の強いタスクをまとめたタスクの集合であり、プロセスを構成する要素である。開発プロセスは、要件定義、方式設計、詳細設計、コード作成、テストなどの作業目的ごとにプロセス内の作業を分割したものである。

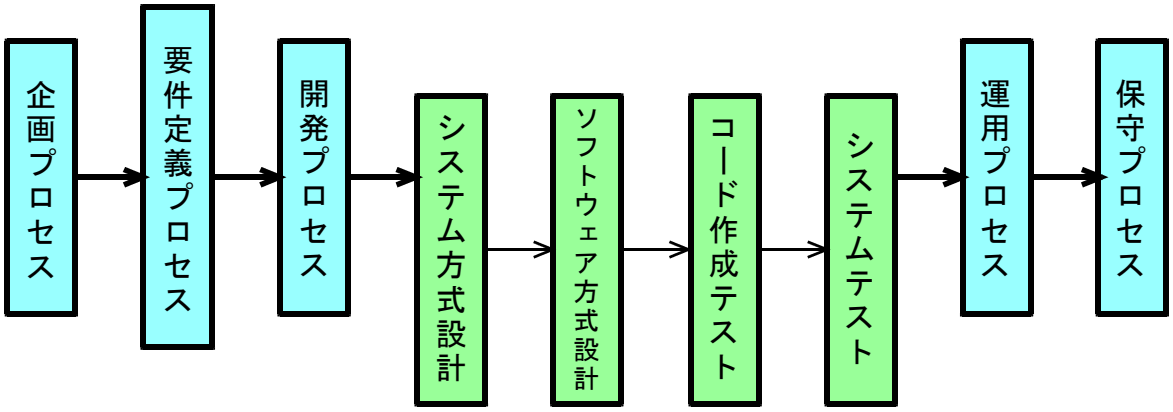
㋒　タスク

タスクはアクティビティを構成する要素である。ソフトウェア方式設計では、コンポーネントの方式設計、インタフェースの方式設計、方式設計の評価、方式設計のレビューなどのアクティビティを具体的に遂行または支援する個々の作業である。

㋓　リスト

タスクを構成する要素である。タスクには多種多様な作業が存在する。

②　システムライフサイクル



プロセス	アクティビティ
企画プロセス	システム化構想の立案、システム化計画の立案
要件定義プロセス	利害関係者要件の定義(業務要件、機能要件、非機能要件の定義)
開発プロセス	●システム方式設計：システム要件の定義、システム方式設計 ●ソフトウェア方式設計：ソフトウェア要件定義、ソフトウェア方式設計、ソフトウェア詳細設計 ●コード作成・テスト：ソフトウェアコード作成、データベース作成、ソフトウェアユニットテスト、データベーステスト ●システムテスト：ソフトウェア結合およびテスト、システム結合およびテスト、システムテストの評価 ●導入・受入：ソフトウェアの導入、受入支援、教育訓練
運用プロセス	運用テスト、業務およびシステムの移行、システム運用、利用者教育、業務運用と利用者支援、システム運用の評価、業務運用の評価投資効果および業務効果の評価
保守プロセス	問題把握および修正分析、修正の実施、保守レビューおよび受入れシステムおよびソフトウェアの廃棄

③　企画プロセス

ⓐ　企画プロセスとは

企画プロセスの目的は、経営事業の目的、目標を達成するために必要なシステムに関係する要求事項の集合とシステム化の方針、及び、システムを実現するための実施計画を得ることである。

ⓑ　システム化構想立案

次のタスクで構成される。

㋐　経営要求・課題の確認
㋑　事業環境・業務環境の調査分析
㋒　現行業務・システムの調査分析
㋓　情報技術動向の調査分析
㋔　対象となる業務の明確化
㋕　業務の新全体像の作成
㋖　対象の選定と投資目標の策定
㋗　システム化構想の文書化と承認
㋘　システム化推進体制の確立

ⓒ　システム化計画の立案

次のタスクで構成される。

㋐　システム化計画の基本要件の確認
㋑　対象業務内容の確認
㋒　対象業務のシステム課題の定義
㋓　対象システムの分析
㋔　適用情報技術の調査
㋕　業務モデルの作成
㋖　システム化機能の整理とシステム方式の策定
㋗　システム化に必要な付帯機能・付帯設備に対する基本方針の明確化
㋘　サービスレベルと品質に対する基本方針の明確化
㋙　プロジェクトの目標設定
㋚　実現可能性の検討
㋛　全体開発スケジュールの作成
㋜　システム選定方針の策定
㋝　費用とシステム投資効果の予測
㋞　プロジェクト推進体制の策定
㋟　経営事業戦略・情報戦略・システム化構想との検証

㋔　システム化計画の作成と承認

㋕　プロジェクト計画の作成と承認

④　開発プロセス

ⓐ　開発プロセスとは

開発プロセスは、仕様やデータ、プログラムなどの入力に対して作業を行い、結果を出力する処理を、ある目的を実現するために複数の処理を順序づけて実行することである。通常、これらの複数の処理の集まりを開発工程という。開発プロセスはシステム開発で実施する処理とその手順を定めたものである。各処理の具体的な作業内容を「アクティビティ」、出力するプログラムやドキュメントなどを「作業成果物」という。

ⓑ　開発プロセスのアクティビティ

㋐　プロセス開始の準備

開発者はプロジェクトの範囲、規模または複雑さに適合したソフトウェアライフサイクルモデルを定義、または選択する。必要な支援プロセス(文書化プロセス、成果物の構成管理プロセスなど)の実施、開発環境の準備、実施計画の作成などを行う。

㋑　システム要件定義

システム要件を明らかにするために、開発するシステムの具体的な利用方法について分析し、システム要求仕様として文書化する。また、システム要件の評価、レビューを実施する。

システム要件では、システム化目標、対象範囲、システムの機能および能力、業務、組織および利用者の要件、信頼性、安全性、セキュリティ、人間工学、インタフェース、操作および保守要件、システム構成要件、設計条件および適格性確認要件、開発環境、品質、コストと期待される効果、移行要件、妥当性確認要件、主要データベースの基本的な要件の定義などを定める。

㋒　システム方式設計

システムの最上位レベルでの方式を確立する。ハードウェア構成品目、ソフトウェア構成品目および手作業を明確にし、システム方式を決め、システム要件を各品目に割り振り、文書化する。システム結合のためのテスト要求事項の定義、システム方式の評価、システム方式設計のレビューを行う。

㋓　ソフトウェア要件定義

ソフトウェア要件の確立、ソフトウェア要件の評価、レビューの実施などを行う。

㋓　ソフトウェア方式設計

ソフトウェア構造とコンポーネントの方式設計、外部・コンポーネント間の各インタフェース方式の設計、データベースの最上位レベルの設計、ソフトウェア結合のためのテスト要求事項の定義、ソフトウェア方式設計の評価、レビューの実施を行う。

㋔　ソフトウェア詳細設計

ソフトウェアコンポーネントの詳細設計、ソフトウェアインタフェースの詳細設計、データベースの詳細設計、ソフトウェアユニットのテスト要求事項の定義、ソフトウェア詳細設計およびテスト要求事項の評価、レビューの実施を行う。

㋕　ソフトウェアコード作成およびテスト

ソフトウェアユニットとデータベースの作成およびテスト手順、テストデータの作成、ソフトウェアユニットとデータベースのテストの実施、ソフトウェアコードおよびテスト結果の評価などを行う。

㋖　ソフトウェアの結合

ソフトウェア結合計画の作成、ソフトウェア結合テストの実施、ソフトウェア適格性確認テストの準備、ソフトウェア結合テストの評価などを行う。

㋗　ソフトウェア適格性確認テスト

ソフトウェア適格性確認テストの実施、評価などを行う。

㋘　システム結合

システム結合計画の作成、システム結合テストの実施、システム適格性確認テストの準備、システム結合テストの評価などを行う。

㋙　システム適格性確認テスト

システム適格性確認テストの実施、システムの評価などを行う。

㋚　ソフトウェア導入

ソフトウェア導入計画の作成、ソフトウェア導入の実施などを行う。

㋛　ソフトウェア受入支援

取得者の受入レビューと受入テストの支援、ソフトウェア製品の納入、取得者への教育訓練および支援などを行う。

⑤　全体計画の立案

ⓐ　トップダウンアプローチ

㋐　企業の目的、戦略という大きな概念から入り、それらを達成するためのあるべき機能、活動、手続といった具合に次第に細部の機能に展開していく方法であり、目的、目標などを機能展開してあるべき姿を描く。
㋑　情報システムの展開も、大きな概念に対応する機能をサポートする情報システムから順次下位のサブ機能をサポートする情報サブシステムへと、上から下に向かって展開していく手法である。
㋒　現状とのギャップを把握し、解決策としての新システム案を作成する。
㋓　理想追求主義、目的指向型と言われる。
㋔　全体計画の立案はトップダウン方式を用いる。

ⓑ　ボトムアップアプローチ

㋐　業務上の現在の問題を中心として、それを解決するシステムを計画する。
㋑　現状を調査し、改善すべき課題を発見する。
㋒　問題点を把握し、改善策としての新システム案を作る。
㋓　現状立脚主義、問題指向型と言われる。
㋔　情報システムの設計、開発にボトムアップ方式を用いる。

ⓒ　ボトムアップ方式と全体計画の問題点

㋐　経営目標や経営戦略の反映よりも、部門目標や部門利益が先行する。
㋑　全社的な課題や長期的な課題よりも、担当者的な視点で当面の問題解決指向になる。
㋒　経営的な視野に立ったリスク評価ができない。
㋓　全社的な効果よりも、局所的な効果が優先する。
㋔　部門による偏りが発生する。
㋕　全社的な調整がしにくい。
㋖　計画遂行時に、経営トップ層や関係部門の一致協力が得にくい。

ⓓ　全体計画立案手順

㋐　立案体制の確立

経営環境、業界動向、競合状況、顧客の動向、経営分析データ、財務状況、各部門の業務達成状況、組織構造などの外部情報、内部情報を収集・整理する。

㋑　経営環境の把握

企業の置かれている状況、全体計画作成の意義、必要性、情報システムの方向、プロジェクトチームへの期待などの説明を受ける。

㋒　業務モデルの定義

企業が遂行する業務全体を知り、業務間の有機的な関連性を理解する。業務モデルを利用して、仕事と情報の関係を情報を使用している仕事、情報を作成している仕事として把握でき、情報システム化の計画に利用できる。

㋓　情報システム体系の定義

情報サブシステム体系の定義とデータベースモデルの定義を行う。情報サブシステム体系は全社レベルの情報システムと情報システムを構成するサブシステムを体系化したものである。データベースモデルは、企業で使用するデータを整理、体系化して論理的なデータベースモデルを構築する。

㋔　情報システムの開発課題の分析

❶　開発の必要性分析

経営戦略上、業務遂行上の観点から情報システムの開発が必要とされている理由を明らかにする。

❷　開発上に存在する阻害要因

開発コストや要員の制約、情報システムを受け入れる企業風土の問題がある。

㋕　情報システム計画の立案

❶　情報サブシステムの開発優先順位の決定

情報システムの効果、経営戦略上の必要性、業務運営上の問題解決効果、組織への影響度などの要素について検討し決定する。

❷　ネットワーク、データベース、ハードウェア、ソフトウェアなど情報システム基盤の整備計画作成

❸　情報システムの開発スケジュールの確立

❹　情報サブシステムの開発工数の見積と要員計画の作成

❺　情報サブシステム開発の費用対効果の分析、回収計画の作成

❻　推進体制の立案

㋖　計画の評価、承認

企画書としてまとめ、内容を評価する。評価は経営的な観点から、合目的性、有効性、投資効果性、開発する場合と開発しない場合のリスクの評価などが対象となる。

⑥　業務モデルの定義

ⓐ　業務モデルとは

㋐　業務モデルの定義

企業活動とその活動に必要な情報、その情報の流れ、データの持ち方を構造化したもので、企業経営のあり方をモデル化したものである。企業における仕事と情報を関連づけて、企業のあるべき姿を写像したものとも言える。

㋑　業務モデルが示すもの

❶　全社全体の業務を示している。
❷　業務で使用する情報とそこで作られる情報を示している。
❸　情報システムの機能と入力情報、出力情報を示している。
❹　業務間のつながりと関連度合いが判別できる。
❺　全社のデータ体系、情報サブシステム間で使用する情報、共用する情報が明確になる。

ⓑ　業務モデルの活用

㋐　情報サブシステムの決定
㋑　データベースモデルの構築
㋒　情報サブシステム間の論理的な開発順序の決定
㋓　業務や組織のあり方の基本

ⓒ　ビジネスプロセス

㋐　ビジネスプロセスは、企業におけるいろいろな業務活動と意思決定活動を、その目的、活動のタイプからグループ分けしたものである。既存の組織や業務手続にとらわれずに、論理的な、あるべき姿を定義する。
㋑　ビジネスプロセスの識別法の一般的な方法は、企業の主要機能分野を識別し、その業務機能を分析してビジネスプロセスを決める要領である。
㋒　企業規模によって、識別する方法が異なり、中小企業の場合は構造化手法、大企業の場合はビジネスエンティティアプローチ法を用いる場合が多い。
㋓　複数の識別手法を組み合わせて実施する場合もある。
㋔　ビジネスプロセスを整理・体系化する場合、業務レベルの大きさ、業務の重要度を考慮して、ビジネスプロセスグループ、ビジネスプロセス、サブビジネスプロセスに体系化・構造化する。
㋕　ビジネスプロセスが体系化されると、ビジネスプロセスの活動内容、使用・作成される情報を明確にし、文書化する。

ⓓ　ビジネスプロセスの識別法

㋐　主要機能を識別し、構造化手法による機能展開を使用する方法

企業の全体的な業務の流れに沿って、主要機能分野を識別し、主要機能分野の業務機能を構造化手法を用いて、大きな機能、１次レベルの機能、２次レベルの機能と順次下位レベルの機能に展開する方法である。

㋑　ＴＱＣの考え方を用いる方法

ＰＤＳの視点から、その分野のビジネスプロセスを識別する方法もある。

㋒　ＤＦＤを使用する方法

主要機能分野のデータフローを作成し、そこから業務機能としてのビジネスプロセスをまとめ、整理する。ビジネスプロセスとデータクラスを同時に定義できる。

㋓　ビジネスエンティティを使用する方法

ＥＲダイヤグラムを使用して、ビジネスエンティティを識別し、ビジネスエンティティの要求、調達、管理、処分などのライフサイクルの視点からビジネスプロセスを識別する。

ⓔ　データクラス

㋐　データクラスは、ビジネスプロセスを遂行するために使用されるデータをユーザが使用する情報の視点からまとめたものである。データをユーザビューから見て、共通の性質をもつクラスにまとめたものと言える。

㋑　データクラスは業務モデルで用いられるものであり、情報システムのモデルではデータベースモデルが使用される。

㋒　データクラスは、ビジネスプロセス、ビジネスプロセス間で使用・生成される情報を明確にすることによって抽出し、入力データ、出力データを整理・統合して設定する。データクラスの識別時に、ＩＰＯダイヤグラムを利用する。

㋓　ＤＦＤでデータフローを作成し、フロー内のデータをまとめてデータクラスを作成する方法もある。

㋔　エンティティ別、データのタイプ別、重要性・情報性の大小などにまとめ、整理・体系化し、文書化する。

ⓕ　業務モデルの作成

㋐　ビジネスプロセスとデータクラスの関連づけ

マトリックス形式で、縦にビジネスクラス、横にデータクラスを配置し、マトリックス上に情報の使用（Ｕ）、作成（Ｃ）の区分を入れる。

㋑　ビジネスプロセス別、情報別の利用状況の明確化

一つの情報がどのビジネスプロセスで活用されているかや、一つのビジネスプロセスがいくつの情報を使用して成り立っているかが分かる。また、情報がどのビジネスプロセスで生成され、加工されているかも分かる。

㋒　表中のビジネスプロセスやデータクラスの項目の配置順は、ビジネス全体の流れや仕事の流れに沿って配列する。

㋓　データクラスの配置は、同一ビジネスプロセスで生成される情報は同じグループにするなどの配慮が必要で、データベース設計のために発生源による情報の入力、管理を明確にしておくことも重要である。

㋔　自社内の業務モデルだけでなく、企業間システム、戦略的情報システムが川下企業、川上企業を含めたビジネス体系で情報システム化を検討する必要がある。

ⓖ　業務モデルの分析

㋐　ビジネスプロセスと現行情報システムとの関連づけ分析

新情報システムを定義する際に、現行システムを評価し、その結果を新システムに反映させる必要がある。マトリックス分析法を適用して評価する。

㋑　ビジネスプロセスと組織の関連づけ

ビジネスプロセスがどの組織で、どのような形で行われているか、意思決定がどのように行われているかを分析し、新しいシステムに反映させる。

㋒　情報システム計画に経営管理者の意見の反映

全体計画立案活動の経過報告、経営及び担当部門の今後の展望の確認、ビジネスプロセス・データクラス・業務モデルのレビュー、現行情報システムの評価、意志決定分野、主要な問題点と解決の方向、情報ニーズとその価値、情報システム化に対する意見・期待などを検討・評価対象にして意見を聞く。

⑦　情報システムモデルの定義

ⓐ　データモデルの定義

㋐　データモデルを構築するには、データクラスをデータの視点から細分化し、見直し、再整理する。この場合、既成のビジネスプロセスにとらわれ、適切なデータモデルが作成されないケースが発生する。

㋑　企業活動を効果的に行うエンティティを識別し、このエンティティについてどのようにデータの管理を行うかを検討し、整理・グルーピングを進めて体系化し、データモデルを作成する。この方法をエンティティアプローチという。

㋒　エンティティが整理された段階で、エンティティとデータクラスを突合せ、相互にチェック、補完し、必要に応じて双方の修正を行う。

ⓑ　情報サブシステムの定義

㋐　情報サブシステムの体系は業務モデルから論理的サブシステムとして導出する。

㋑　個々の情報サブシステムは、業務モデルに表示されているビジネスプロセスとデータクラスとの関わり方、仕事のやり方、システムの利用方法、システムの規模などを考慮して、全体の中の部分として割り付けられる。

㋒　個々の情報サブシステムを開発する際には、論理的サブシステムの枠組みの中での位置づけを明確にし、計画的にシステム構築を積み重ねる。

例題演習

情報戦略における全体最適化計画策定の段階で，業務モデルを定義する目的はどれか。

ア　企業の全体業務と使用される情報の関連を整理し，情報システムのあるべき姿を明確化すること

イ　システム化の範囲や開発規模を把握し，システム化に要する期間，開発工数，開発費用を見積もること

ウ　情報システムの構築のために必要なハードウェア，ソフトウェア，ネットワークなどの構成要素を洗い出すこと

エ　情報システムを実際に運用するために必要な利用者マニュアルや運用マニュアルを作成するために，業務手順を確認すること

解答解説

全体最適化に関する問題である。

システムや組織において、各部分機能の最適を図ることを部分最適、システム・組織の全体の最適を図ることを全体最適という。全体最適化は、企業及び企業グループで調達、生産、物流、販売など個々の業務機能のみの生産性を上げる(部分最適)のではなく、業務機能全体の効率や生産性を最適化する（全体最適）ことを前提に､企業や企業グループの収益を最大化する目的に使われる。全体最適化計画の策定はシステム化構想の立案時に検討される。

アの全体業務と情報システムのあるべき姿の明確化はシステム化構想の立案段階に行われる内容であり、業務モデルを定義することによって可能となる。求める答えはアとなる。

イのシステム化の開発期間、工数、費用の見積もりは、システム化計画の立案で行われる。

ウの情報システムの構成要素の洗い出しは、システム化計画の立案で行われる。

エのユーザマニュアルや運用マニュアルの作成準備は運用プロセスで行われる。

例題演習

共通フレームによれば，システム化構想の立案で作成されるものはどれか。

ア　企業で将来的に必要となる最上位の業務機能と業務組織を表した業務の全体像
イ　業務手順やコンピュータ入出力情報など実現すべき要件
ウ　日次や月次で行う利用者業務やコンピュータ入出力作業の業務手順
エ　必要なハードウェアやソフトウェアを記述した最上位レベルのシステム方式

解答解説

システム化構想立案に関する問題である。

企画プロセスのシステム構想立案時に業務の新全体像を作成する。企業で将来必要となる最上位の業務機能と業務組織のモデルを検討する。検討の結果、目標とする業務の新しい全体像を描き、新システムの全体イメージを作成し、業務機能と組織モデル、新システムとが整合しているかを確認する。

アは企画プロセスのシステム化構想立案時に作成、イ、エはシステム方式設計時、ウはソフトウェア方式設計時で作成され、開発プロセスで行う作業である。求める答えはアとなる。

例題演習

ソフトウェアライフサイクルを，企画，要件定義，開発，運用，保守のプロセスに区分したとき，企画プロセスの目的はどれか。

ア　新しい業務の在り方や運用をまとめた上で，業務上実現すべき要件を明らかにすること
イ　事業の目的，目標を達成するために必要なシステムに関係する要求事項の集合とシステム化の方針，及びシステムを実現するための実施計画を得ること
ウ　システムに関する要件について技術的に実現可能であるかどうかを検証し，システム設計が可能な技術要件に変換すること
エ　システムの仕様を明確化し，それを基にＩＴ化範囲とその機能を具体的に明示すること

解答解説

ソフトウェアライフサイクルにおける企画プロセスの目的に関する問題である。

企画プロセスの目的は、経営事業の目的、目標を達成するために必要なシステムに関係する要求事項の集合とシステム化の方針、および、システム化を実現するための実施計画を得ることである。システムが関与する経営戦略を受けて、システム化構想の立案、システム化計画の立案に関わる活動が行われる。

アの業務のあり方、入出力情報などは要件定義プロセスで行う。

イの経営事業の目的、目標を達成するために必要なシステムに関係する経営上のニーズ、システム化の方針、システム化実施計画などは、企画プロセスで行われる。求める答えはイとなる。

ウの必要なシステム機能、システム開発方式などは開発プロセスで行われる。
エの必要なシステムの機能、能力、ライフサイクルなどは開発プロセスで行われる。

例題演習

共通フレームによれば，企画プロセスの目的はどれか。

ア　経営事業の目的，目標を達成するために必要なシステム化の方針及びシステムを実現するための実施計画を得る。
イ　作業成果物及びプロセスが，定義された条件及び計画に従っていることを保証する。
ウ　発見されたすべての問題を，識別，分析，管理及び制御して，解決することを確実にする。
エ　プロセスによって生成され記録されたシステム又はソフトウェア情報を文書化し，保守する。

解答解説

企画プロセスに関する問題である。
企画プロセスの目的は、経営事業の目的、目標を達成するために必要なシステムに関係する要求事項の集合とシステム化の方針、および、システムを実現するための実施計画を得ることである。求める答えはアとなる。
アは主ライフサイクルプロセスの企画プロセス、イ、ウ、エは支援ライフサイクルプロセスのイは検証プロセス、ウは問題解決プロセス、エは文章化プロセスである。

例題演習

共通フレームによれば，企画プロセスで定義するものはどれか。

ア　新しい業務の在り方や業務手順，入出力情報，業務を実施する上での責任と権限，業務上のルールや制約などの要求事項
イ　業務要件を実現するために必要なシステムの機能や，システムの開発方式，システムの運用手順，障害復旧時間などの要求事項
ウ　経営事業の目的，目標を達成するために必要なシステムに関係する経営上のニーズ，システム化，システム改善を必要とする業務上の課題などの要求事項
エ　求められているシステムを実現するために必要なシステムの機能，能力，ライフサイクル，信頼性，安全性，セキュリティなどの要求事項

解答解説

共通フレームによる企画プロセスの定義に関する問題である。
企画プロセスの目的は、経営事業の目的、目標を達成するために必要なシステムに関係する要求事項の集合とシステム化の方針、および、システム化を実現するための実施計画を得ることである。システムが関与する経営戦略を受けて、システム化構想の立案、システム化計画の立案に関わる活動が行われる。
アの業務のあり方、入出力情報などは要件定義プロセスで行う。

イの必要なシステム機能、システム開発方式などは開発プロセスで行われる。

ウの経営事業の目的、目標を達成するために必要なシステムに関係する経営上のニーズなどは、企画プロセスで行われる。求める答えはウとなる。

エの必要なシステムの機能、能力、ライフサイクルなどは開発プロセスで行われる。

例題演習

共通フレームによれば，企画プロセスで実施すべきものはどれか。

ア　新しい業務の在り方を整理し，業務プロセスや業務ルールを明確にする。
イ　新しく開発されるシステムへの移行時期及び移行手順を明確にする。
ウ　業務の新しい全体像及び新システムの全体イメージを作成する。
エ　ソフトウェアユニットのテスト要求事項及び予定を定義する。

解答解説

企画プロセスに関する問題である。

企画プロセスは、経営要求・課題の確認、事業環境・業務環境の調査分析、現行業務・システムの調査分析、情報技術動向の調査分析、対象となる業務の明確化、業務の新全体像や新システムの全体イメージの作成、対象の選定と投資目標の策定、システム化構想の文書化と承認、システム化推進体制の確立などのタスクで構成される。

アは開発プロセスのシステム要件の定義、イは運用プロセスの業務システムの移行、ウは企画プロセスのシステム化構想の立案、エは開発プロセスのソフトウェアテストである。求める答えはウとなる。

例題演習

システム開発の最初の工程で行う作業として，適切なものはどれか。

ア　各プログラムの内部構造を設計する。
イ　現状の業務を分析し，システム要件を整理する。
ウ　サブシステムをプログラム単位まで分割し，各プログラムの詳細を設計する。
エ　ユーザインタフェースを設計する。

解答解説

システム開発工程に関する問題である。

システム開発工程で最初に行う作業はモデリングのためのシステム要件を明確にするために、現状業務を分析することである。従って、現状業務を分析し、システム要件を整理することになる。求める答えはイとなる。

システム要件整理後、モデリングを行い、その後モデルに従って、システムの構築、評価の順に展開される。モデリングの段階で、システム分割、コンポーネント設計、ユーザインタフェース設計が行われ、システム構築段階にプログラムの内部構造が設計される。

例題演習

共通フレームによれば，要件定義プロセスの活動内容には，利害関係者の識別，要件の識別，要件の評価，要件の合意などがある。このうち，要件の識別において実施する作業はどれか。

ア　システムのライフサイクルの全期間を通して，どのエ程でどの関係者が参画するのかを明確にする。
イ　抽出された要件を確認して，矛盾点や曖昧な点をなくし，一貫性がある要件の集合として整理する。
ウ　矛盾した要件，実現不可能な要件などの問題点に対する解決方法を利害関係者に説明し，合意を得る。
エ　利害関係者から要件を漏れなく引き出し，制約条件や運用シナリオなどを明らかにする。

解答解説

要件定義プロセスに関する問題である。

要件定義プロセスの目的は、新たに構築する業務、システムの仕様を明確化し、それをベースにＩＴ化範囲とその機能を明示することである。活動内容として、利害関係者の識別、要件の識別、要件の評価、要件の合意などがある。

アは利害関係者の識別、イは要件の評価、ウは要件の合意、エは要件の識別である。求める答えはエとなる。

例題演習

要件定義の段階で行う作業はどれか。

ア　新たに構築する業務とシステムの仕様を明確化し，システム化範囲を明示する。
イ　顧客が記述したニーズに合ったソフトウェアを開発する。
ウ　事業の目的，目標を達成するために必要なシステム化の方針を立案する。
エ　ソフトウェア製品の運用及び利用者に対する運用支援を行う。

解答解説

要件定義で行う作業に関する問題である。

要件定義は、システム化要件を定義することである。次の内容について要件を定義する。

① システム化の目標と範囲
② システム化の機能要件・性能要件
③ 業務処理手順
④ 入出力情報要件
⑤ 操作要件
⑥ 実行環境要件
⑦ 周辺インタフェース要件
⑧ 開発環境要件
⑨ 主要データベース・主要データ項目に関する要件

アはシステム化要件定義であり、要件定義プロセスで行う。
イはソフトウェアの開発であり、開発プロセスで行う。
ウはシステム化方針の立案であり、企画プロセスで行う。
エはシステムの運用、運用支援であり、運用プロセスで行う。

例題演習

要件定義プロセスで実施すべきものはどれか。

ア　新しい業務の手順やルール，制約条件を明確にし，利害関係者間で合意する。
イ　新システムによる業務運用の投資効果及び業務効果の実績を評価する。
ウ　法規制，経済状況などの事業環境を分析し，事業目標や業務目標を作成する。
エ　要求事項を満たしているか，ソフトウェア及びデータベースのテストを実施する。

解答解説

要件定義で行う作業に関する問題である。
アはシステム化要件定義であり、要件定義プロセスで行う。
イの費用と投資効果は、企画プロセスのシステム化計画の立案で行う。
ウの事業環境の分析、事業目標、業務目標の作成は、企画プロセスのシステム化構想の立案で行う。
エのソフトウェア、データベースのテストは開発プロセスで行う。

例題演習

システム化計画を立案するときに考慮すべき事項はどれか。

ア　運用を考えて，自社の社員が開発する前提で検討を進める。
イ　開発，保守，運用に関する費用と投資効果を明確にする。
ウ　失敗を避けるため，同業他社を調査し，同じシステムにする。
エ　テスト計画，運用マニュアル及び障害対策を具体的に示す。

解答解説

システム計画を立案する場合の考慮事項に関する問題である。
共通フレームは、コンピュータ・システムの開発において、システム発注側と受注側の間で相互の役割や業務の範囲・内容、契約上の責任などに対する誤解がないように、双方に共通して利用できるよう用語や作業内容を標準化するために作られたガイドラインである。システム構築・運用の受発注において、契約上のトラブル防止、作業内容の確認、役割分担の明確化、社内作業標準の策定や人員計画、見積もり精度の向上、品質確保などに利用される。
アの場合、運用の方法や運用に関係する開発関連業務において、他社との関係やユーザ・ベンダーの関係が発生することがあり、自社の社員だけで開発することを前提に検討することは好ましくない。
イの開発、保守、運用に関する費用と投資効果を明確にする作業はシステム化計画立案時に

行う。求める答えはイとなる。

ウの同業他社を調査しても、それぞれの企業の特徴、業務上の問題、プロジェクト上の特徴が異なり、必ずしも同じシステムにはならない。参考にする程度になる。

エのシステム計画の段階で、運用マニュアル、障害対策を具体的にすべて示すことは不可能に近い。通常は不能である。

例題演習

非機能要件の定義に該当するものはどれか。

ア　業務を構成する機能間の情報（データ）の流れを明確にする。
イ　システム開発で利用する言語に合わせた開発基準，標準を作成する。
ウ　システム機能として実現する範囲を定義する。
エ　他システムとの情報授受などのインタフェースを明確にする。

解答解説

要件定義の非機能要件に関する問題である。

業務要件を実現するために必要なシステムの機能要件以外の要件を非機能要件という。

非機能要件は、品質要件、技術要件、運用・操作要件、移行要件、付帯要件などが含まれる。

技術要件には、システムの実現方法、システム構成、システム開発方式、開発基準、標準、開発環境などが含まれる。

機能要件は業務要件を実現するために必要なシステム要件であり、業務内容(手順、入出力情報、組織、責任、権限など)、業務特性(ルール、制約など)、業務用語、外部環境と業務の関係・授受する情報(インタフェース)などが含まれる。

ア、ウ、エの内容は機能要件であり、イの内容の開発基準、標準は非機能要件である。求める答えはイとなる。

例題演習

非機能要件定義を説明したものはどれか。

ア　業務要件のうち，システムで実現が難しく，手作業となる業務機能を明確化する。
イ　業務要件の実現に必要な，品質要件，技術要件，運用要件などを明確化する。
ウ　業務要件を確定させるために，現行システムで不足している機能を明確化する。
エ　業務要件を実現するために，新たに導入するパッケージの適合性を明確化する。

解答解説

要件定義の非機能要件に関する問題である。

業務要件を実現するために必要なシステムの機能要件以外の要件を非機能要件という。

非機能要件は、品質要件、技術要件、運用・操作要件、移行要件、付帯要件などが含まれる。

技術要件には、システムの実現方法、システム構成、システム開発方式、開発基準、標準、開発環境などが含まれる。

機能要件は業務要件を実現するために必要なシステム要件であり、業務内容(手順、入出力情報、組織、責任、権限など)、業務特性(ルール、制約など)、業務用語、外部環境と業務の関係・授受する情報(インタフェース)などが含まれる。

ア、ウ、エは、機能要件であり、イは非機能要件である。求める答えはイとなる。

例題演習

ＩＴ投資評価を，個別プロジェクトの計画，実施，完了に応じて，事前評価，中間評価，事後評価として実施する。事前評価について説明したものはどれか。

ア　事前に設定した効果目標の達成状況を評価し，必要に応じて目標を達成するための改善策を検討する。

イ　実施計画と実績との差異及び原因を詳細に分析し，投資額や効果目標の変更が必要かどうかを判断する。

ウ　投資効果の実現時期と評価に必要なデータ収集方法を事前に計画し，その時期に合わせて評価を行う。

エ　投資目的に基づいた効果目標を設定し，実施可否判断に必要な情報を上位マネジメントに提供する。

解答解説

ＩＴ投資の事前評価に関する問題である。

事前評価は、企画プロセスで作成されるシステム化構想やシステム化計画に対して実施される評価であり、経営戦略に対する合目的性、投資効果性、実現可能性、開発の合理性、運用の容易性、技術的整合性などの項目の内容の妥当性が評価される。

投資目的に基づいた効果目標を設定し、実施可否判断に必要な情報を上位マネージメントに提供する。

ア、イは中間評価、ウは事後評価、エは事前評価の説明である。求める答えはエとなる。

txt040112 開発モデルと構造化手法

① ウォータフォールモデル

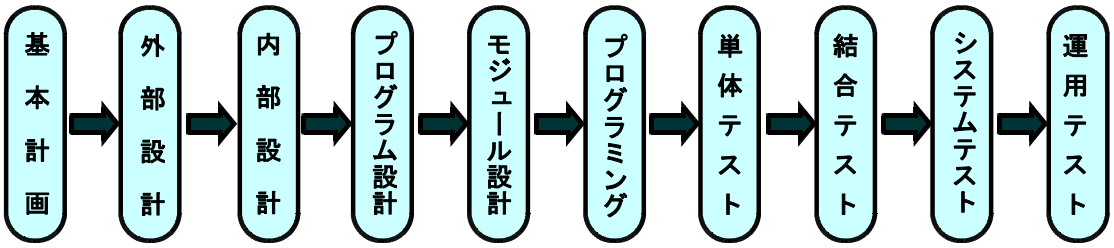
ⓐ ウォータフォールモデルとは

ウォータフォールモデルは、ソフトウェアの作成から廃棄までのシステムライフサイクルプロセスを、いくつかの工程に分割し、前工程の出力を次工程の入力にする方式である。ソフトウェアの開発工程を区切りながら上流から下流に向けて、基本計画、外部設計、内部設計、プログラム設計、プログラミング、テストと順次、作業を進める開発プロセスモデルである。

ⓑ 各工程の作業内容

工程名称	作業内容
基本計画	システム計画の作成、開発すべきシステムの要求定義を行う。
外部設計	要求分析を基にシステム機能を確定し、機能を実現するためのシステム構成を明確化し、論理データ設計、インターフェース設計、コード設計などを行う。
内部設計	システム構築上必要な機能をプログラムに分割し、プログラム間の処理を明確化し、プロセスフローを作成する。
プログラム設計	各プログラムの構造設計、モジュール分割、各モジュール間のインタフェースを決定する。
モジュール設計	モジュール内の詳細処理手順を設計する。論理設計を行う。
プログラミング	各モジュールのコーディングを行う。
単体テスト	モジュール単体としての稼働を確認する。
結合テスト	モジュール間の結合テストを行い、プログラムとしての稼働を確認する。
システムテスト	システムの機能、性能、操作性、障害管理などを総合的にテストして、システム全体としての稼働を確認する。
運用テスト	本番と同じシステム環境やデータで稼働を確認する。新システムが有効に活用できるや否やを確認する。

ⓒ 工程の流れ



ⓓ　ウォータフォールモデルの特徴

㋐　計画、設計、プログラミング、テストと一定の方向に向けて、各工程を区切り、成果物を確認しながら段階的に開発を進める。逐次的な処理手順は工程全体を見通しやすくする。
㋑　大規模システムの開発に適しているが、開発費用や工数が多くなる。
㋒　各工程の最後には必ず検証が組み込まれる。徹底的に検証し、次工程にバグを持ち込まない原則で進める。
㋓　次工程の作業実施中に間違いを発見しても前工程しか戻れない。基本的には後戻りが許されないため、環境変化やユーザニーズの変更による仕様変更や設計変更が困難であり、開発体制の柔軟性に問題がある。
㋔　開発工程の最初の段階で、システム要求定義およびソフトウェア要求定義を決定しなければならない。

ⓔ　ウォータフォールモデルの問題点

㋐　トップダウン的な作業になるため後戻りは原則として許されず、恒常的に発生する仕様変更の要求などエンドユーザの多種多様な要求に対応できない。
㋑　要件定義段階で、すべての要件を洗い出すことが困難である。上流工程に問題があってもプロジェクトの終盤まで発見できないケースが多い。
㋒　システム開発全体に時間がかかりすぎ、バックログの増大を招く危険性がある。
㋓　既存システムの再利用が難しい。

②　スパイラルモデル

ⓐ　スパイラルモデル

ウォータフォールモデルにプロトタイプモデルの手法を取り入れた開発モデルで、一方向を維持しながら、何回も繰り返す手順をらせん状に実現する。目標設定の最適化フェーズ、要求定義の分析開発フェーズ、レビューの検証フェーズ、ソフトウェア開発の計画フェーズで１サイクルを完了する。再分析、設計、プログラミング、テストとサイクルをあがるように移動する開発手法である。

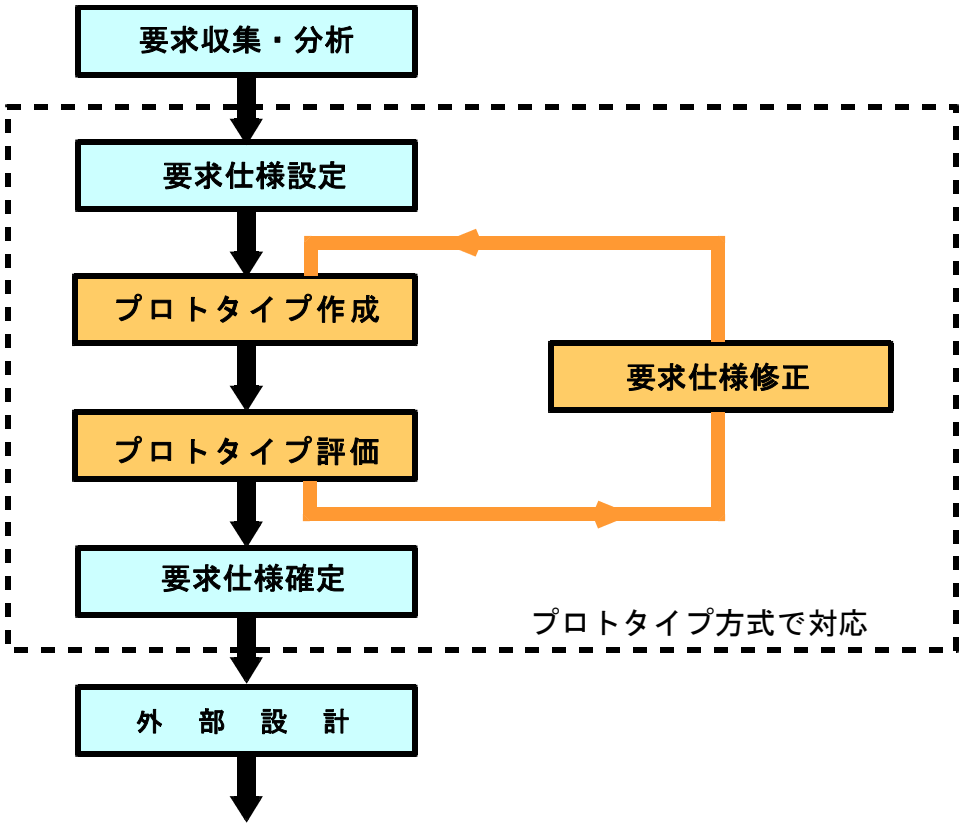
ⓑ　スパイラルモデルの特徴

㋐　検証フェーズでフィードバックの機構が実現し、仕様の変更に柔軟に対応する。
㋑　プロトタイピングによるソフトウェア開発に適応できる。
㋒　顧客の要求を早い段階に確認することやシステムの実現を事前に予測し、システムの定義と実装、確認と評価を繰返ながら拡大、詳細化するためリスクが最小化される。

㊓　稼働の容易さ、導入の効率性、変更の迅速性、進化性の特性がある。
㊔　機能を順次追加しながら短期間で開発することができる。

③　プロトタイプモデル

ⓐ　プロトタイプモデル



ウォータフォールモデルの難点を解決するために考えられたモデルで、ユーザの要求仕様を開発の早い段階に目に見える試作品として現実化する手法である。ユーザの要求仕様の内容が的確に反映できるため手戻りの少ないシステム開発が可能になる。

ⓑ　プロトタイプモデルの特徴

㊀　問題点が早期に発見でき、短期間でシステム開発を完了する。
㊁　ユーザの意向を反映しながら開発を進めるため、ユーザ部門の参加意識が高くなる。
㊂　開発者に多方面の知識や技術が要求される。
㊃　小規模システムには適しているが、改良・発展させると継ぎ接ぎのシステムとなり、整合性の悪い効率の低いシステムになりやすい。

④　インクリメンタルモデル

ⓐ　インクリメンタルモデル

システムを独立性の高いいくつかのサブシステムに分割して、サブシステムごとに順次開発、リリースしていくプロセスモデルであり、サブシステムの開発が並列進行する点が、スパイラルモデルと異なる。開発の各ステップをプロトタイピングのように繰り返し行う手法で、各ステップをずらして実行し、それぞれ追加機能を開発する。最初に開発した部分は基本的な要求事項を満たしたコア製品で、追加すべき機能は顧客に実際に利用してもらってその結果をレビューし、修正や追加機能の開発を進めていく。市販のワープロソフトや表計算ソフトなどの汎用ソフト開発に利用されている考え方である。

ⓑ　インクリメンタルモデルの特徴

㋐　反復型であるが、追加作業が終わるたびに製品として納入できる。
㋑　ユーザに対していくつかの主要な機能を提供し、評価を可能にする。
㋒　開発要員の手配が必要に応じて増減可能である。
㋓　技術上のリスクを有効に管理できる。

⑤　オブジェクト指向モデル

ⓐ　オブジェクト指向モデルとは

クラスの考え方の利用により、開発の生産性向上を図ることを目標とする開発手法である。データとデータ操作に用いるアルゴリズムの両方をカプセル化するクラスの作成が重要で、クラスを部品として再利用することによって効率的な仕様変更に柔軟に対応できるシステム開発が可能になる。プログラミングを意識する必要が少なく、ユーザがシステム化対象業務の実体をそのままシステムとして実現させるような開発も可能となる。

ⓑ　オブジェクト指向型開発の長所

㋐　データとプロセスを必ずセットにして考える。
㋑　標準ライブラリを設ければ、既存システムの再利用が十分可能である。
㋒　ユーザの要求に対して、リアルタイムに対応可能で、後工程での修正に対応しやすい。
㋓　ユーザ主導による開発も可能である。

⑥　アジャイル開発

ⓐ　アジャイル開発とは

一定の周期でソフトウェアをつくり、それを発展させて開発を進める反復的な開発アプローチである。アジャイル宣言の考え方に基づいて、顧客のニーズの変化に即応できる開発を、機動性に富んだチームと個人の自律性を活用して実施する。ソフトウェアの早期、継続的な納品により、顧客の満足を達成することを優先する。要求内容の変更に常に即応でき、数週間から数ヶ月のサイクルで動作するソフトウェアを納入する。サイクルは短い方がよい。アジャイル開発手法には、ＸＰ、スクラム、ＤＳＤＭなどがある。

ⓑ　アジャイル宣言

㋐　プロセスやツールよりも個人や相互作用を活用する。
㋑　わかりやすいドキュメントよりも動くソフトとウェアを求める。
㋒　契約上の駆け引きよりも顧客とのコラボレーションを重視する。
㋓　計画を硬直的に守るよりも変化に対応する。

⑦　開発プロセスの選択

ⓐ　開発プロセスの選択基準

開発対象となるシステムやプロジェクトの特性、スケジュール、開発体制、採用する技術、要員のスキルなどに応じて、開発プロセスを選択あるいは組み合わせて利用する。プロジェクトをタイプ別に分け、開発プロセスを選択する際の考慮点をまとめると次のようになる。

㋐　大規模プロジェクト

大規模プロジェクトではプロジェクトを実行可能な単位に分割することが重要なポイントになる。作業を分割することで管理対象を小さくする。サブシステムに分割する前に、要件定義を確実に実施する。要件定義が不十分な場合、サブシステム間のインタフェースの不整合やデータベースの重複・不整合などが発生する。要件定義では、開発範囲を明確にする。

要件定義の完了後に、サブシステム分割を行い、サブシステムの特性や開発期間などに応じて、適切な開発プロセスを選択する。リスクを伴うシステムには反復型、そうでなければウォータフォール型を使用する。

㋑　Ｗｅｂシステム開発

新たな要求が次々に出てきたり、要求自体が変化することが多い短納期のＷｅｂシステム開発では、短いサイクルで反復を繰返し、順次リリースできる反復型を適用する。反復型であれば、設計変更が発生してもその影響を該当する反復サイクルだけにとどめられる。

㋒　パッケージ型開発

ＥＲＰやＳＣＭ、ＣＲＭなどのパッケージソフトを導入する場合、「フィット・アンド・ギャップ分析」が必要になる。フィットは、システムのあるべき姿がパッケージソフトの標準機能でどれだけ実現できるかを分析することである。ギャップは実現できない機能を明確にすることである。ギャップの部分は追加開発しなければならない。開発コストや開発期間を抑えるために、どれだけ追加開発を少なくできるかがポイントになる。業務プロセスをパッケージソフトの機能に合わせて変更することも必要になる。

分析結果を踏まえて、要件を洗い出す。フィット部分についてはパラメータの設定やベンダーの提供するツールで対応する。ギャップ部分は追加開発する。追加開発の部分は通常のシステム開発と同様なアプローチを用いる。ウォータフォール型や反復型などの開発プロセスを必要に応じて適用する。

ⓑ　開発プロセス選択の要素

㋐　開発プロジェクトのサイズと期間

要員数、開発期間など

㋑　開発システムの特性

システムの重要度、システム障害が引き起こす結果の大きさ、ビジネス・アプリケーション／科学技術計算／組み込みシステムなどの対象、システムの複雑性など

㋒　開発体制、開発環境

ユーザ・協力会社を含めた開発体制、開発ロケーション(分散／集中)、オフショア開発の有無、適用する開発技術／コンポーネント／パッケージソフトなど

㋓　使用する開発支援ツール

ツールで自動化できる範囲、特定の開発プロセス／手法の適用が前提か否かなど

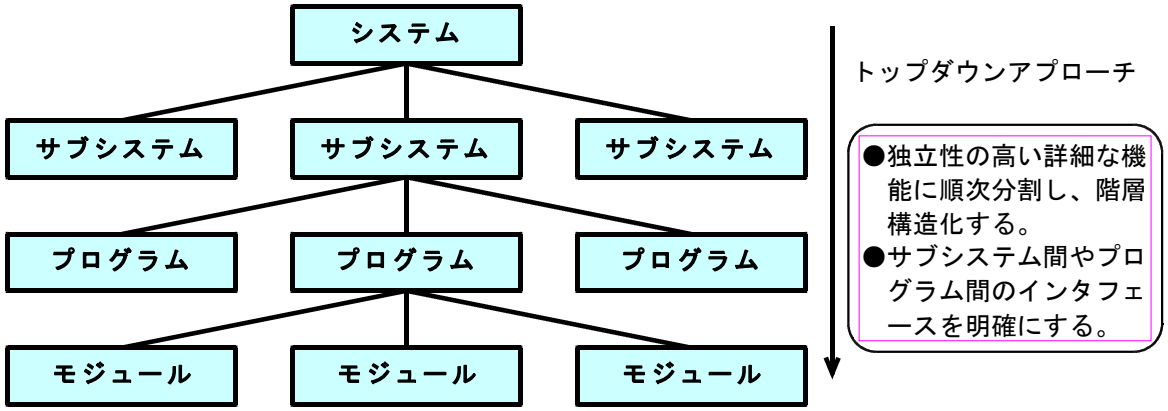
㋔　要員のスキル・経験

業務知識、開発プロセス・開発手法の知識、開発ツールの知識など

⑧　機能分割構造化

ⓐ　トップダウンアプローチの考え方

トップダウンアプローチは、システムの上位レベルから下位レベルに向かって詳細化を行う方法である。システムをサブシステムに、サブシステムを更に詳細化して、プログラムレベルやモジュールレベルまで段階的に詳細化する考え方である。



ⓑ　トップダウンアプローチの期待効果

㋐　機能の抜けがなくなる。
㋑　必要機能とシステム全体との関連づけが明確になる。
㋒　内部設計以降のプログラム設計とのつながりが深くなる。

ⓒ　トップダウンアプローチの実行手順

㋐　外部環境の明確化

システム全体の外部環境、入出力データの明確化を行う。

㋑　サブシステムへの分割

システム全体を詳細な構成要素であるサブシステムに分割する。サブシステム間の関係を定め、サブシステム間のインタフェースを明らかにする。

㋒　サブシステムの詳細化

サブシステムを更に詳細な機能をもつものに分割する。詳細化はプログラム単位まで行う。呼び出す機能が多くなりすぎると関連する機能をまとめる。分割の目安は３～１０個程度である。

ⓓ　機能分割構造化設計

機能分割構造化は、システムをサブシステムからプログラムレベルに、トップダウンアプローチの考え方に基づいて分割することである。システムフロー、サブシステム構成図、サブシステム関連図をもとに、階層化されたデータフローダイヤグラム（ＤＦＤ）などの図的表現手段を活用して、プロセスをプログラムやモジュールレベルまで分割する。機能分割・階層構造化は、機能の独立性、機能の強度と結合度に着目して行う。

ⓔ　機能分割・構造化の手順

㋐　機能の洗い出し

システムの機能を整理し、システムの機能をすべて洗い出す。

㋑　データフローの明確化

データがシステムの中で変換される順序に従い、機能を関連づける。表現手段として、階層化されたデータフローダイヤグラム（ＤＦＤ）が用いられる。

㋒　機能のグループ化

洗い出した機能を処理内容に応じてグループ化する。

㋓　階層構造化

機能レベルを階層構造で示す。表現手段として階層構造図が用いられる。

㋔　プログラム機能の決定

機能を段階的に分割すると、最下位レベルのプログラムとなる。階層構造化の結果を受けて、プログラムの処理内容を決定する。

㋕　プログラム間インタフェースの明確化

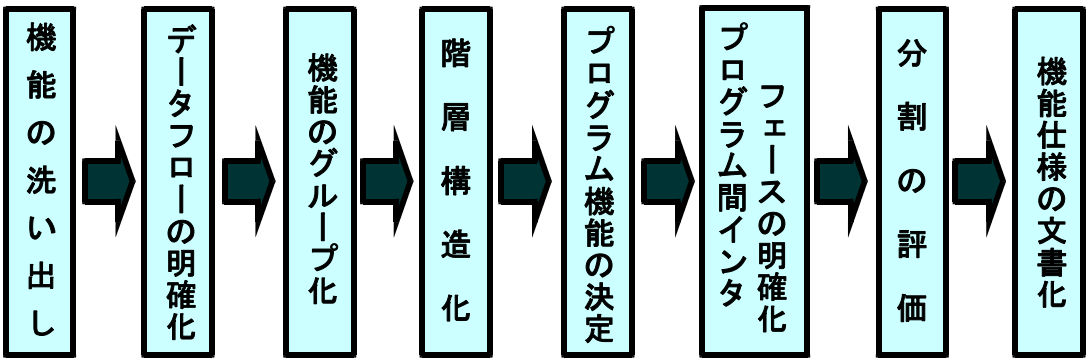
出力データに注目し、関係する入力データや処理内容を決定する。１つのプログラムの出力データがプログラム間のインタフェースとなる。

㋖　分割の評価

階層構造化を行った機能に対して評価を行う。

㋗　機能仕様のドキュメント化

プログラム単位に分割された機能をドキュメント化する。



ⓕ　機能分割評価の指針

㋐　機能の強度

強度は機能内部の各構成要素の関連性を示す尺度である。強度が強い機能は独立性が高い。

㋑　機能の結合度

結合度は各機能間の関連の強弱を示す尺度である。結合度が弱いほど独立性が高い。

ⓖ　作成ドキュメント

㋐　サブシステム構成図

サブシステム機能とそれを実現するためのプログラム構造、各プログラムの目的と機能、入出力データの関連を表す。

㋑　プログラム間インタフェース

プログラム相互に関連するインタフェース情報について記述する。

㋒　プログラム機能

処理内容、処理方法、エラー処理、メッセージ一覧、入出力データ、制約事項などを記述し、文書化する。

⑨　プログラム構造化設計

ⓐ　プログラム設計の問題点

㋐　設計方法が必ずしも徹底していない。
㋑　一般的にプログラム設計の時間が少ない。

ⓑ　良い設計のポイント

㋐　複雑さの減少
㋑　適切な大きさに分割
㋒　独立性の実現
㋓　階層構造化

ⓒ　構造化設計の手順

㋐　最上位モジュールの定義

機能分析に必要な全体的な機能を定義する。

㋑　モジュールの機能分析

上位モジュールで定義した機能を分析し、それを実行するために必要な下位の機能を明らかにする。

㋒　分割技法の選択

モジュールを分割するための最適な分割技法を選択する。ＳＴＳ分割技法、ＴＲ分割技法、共通機能分割技法、ジャクソン法、ワーニエ法などの分割技法が検討対象になる。

㋓　モジュール分割、階層構造化

モジュールに分割し、階層構造図を作成する。

㋔　モジュール間インタフェースの定義

上位モジュールと直接従属モジュールとのインタフェースを明確にする。

㋕　更に分割すべきモジュールを調べる。必要な場合には①に戻る。

⑩　構造化設計手法

ⓐ　流れ図

㋐　流れ図とは

どのような手順で処理を実行するかを図で表示したものである。流れ図で使用できる記号はＪＩＳで規定されている。

㋑　流れ図の特徴

❶　処理の手順に従って記述されているため、処理手順の誤りなどを容易に発見できる。
❷　プログラムの経験がなくても使用することが可能である。
❸　ファイルやデータの受け渡しが明確になる。

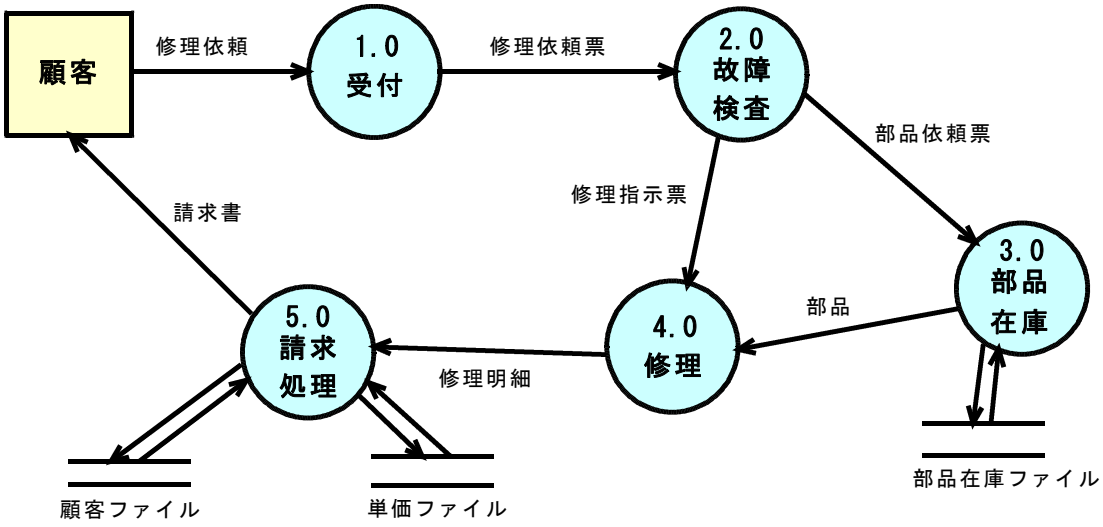
㋒　流れ図の種類

データ流れ図は問題解決におけるデータの経路を表し、使用する各種のデータ媒体とともに処理手順を定義したものである。プログラム流れ図はプログラム中における一連の処理手

順を示したものである。システム流れ図はシステムのデータに対する処理やデータの流れを表現したものである。

ⓑ　ＤＦＤ

㋐　データフローダイヤグラム（ＤＦＤ）とは



業務処理におけるデータの流れを図的に表現したものである。データの源泉・吸収、データフロー、データ蓄積、処理の記号からなる。

㋑　ＤＦＤの特徴

❶　理解しやすい。

記号の種類が少ないので分かりやすい。

❷　階層化が可能である。

概要設計レベルから詳細設計レベルまで、種々のレベルに対応した階層的な記述ができる。

❸　データ中心である。

データの流れに注目して記述しているため、どのデータがどこで発生し、どのような処理によって変換されたかということが明確に表現できる。

❹　分析から設計まで使用可能である。

システムの分析から設計段階まで使用することができるため、同一手法の利用によって生産性が向上し、コミュニケーションの手段としても効果的である。

㋒　ＤＦＤの作成手順

❶　最上位のＤＦＤであるコンテキストダイヤグラムを作成する。

コンテキストダイヤグラムは単一プロセスとして記述し、システムの基本的な機能を表す。その機能を実現するための源泉データ、吸収データを決定する。

❷ **バブルの段階的な詳細化を図る。**

各レベルのバブルを分割したものを、下位のレベルでのバブルとして記述し、各バブル間のデータの流れを決定する。上位のＤＦＤとの整合性を保たなければならない。

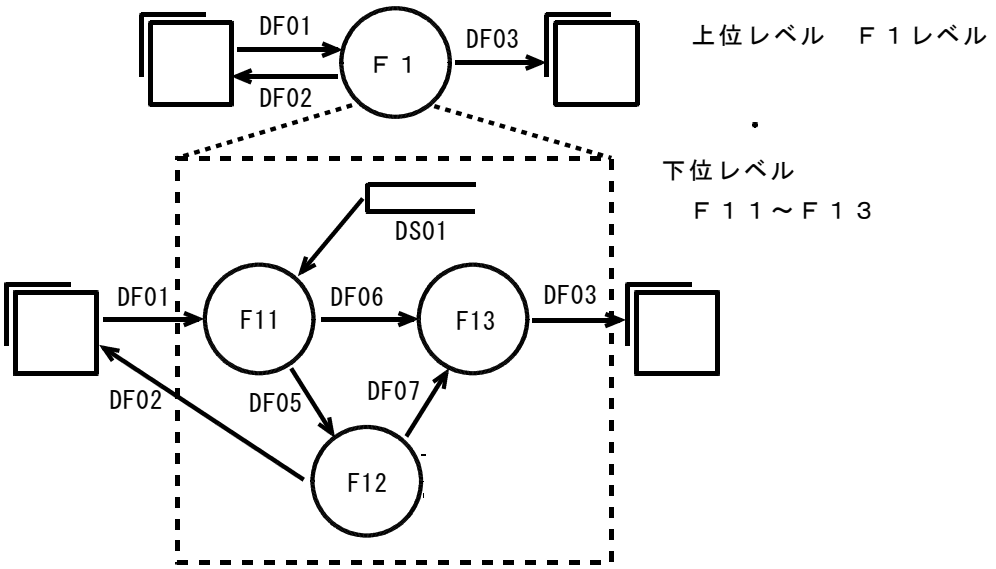
❸ **最下位レベルのＤＦＤに記載された各バブルごとにそれぞれのミニ仕様を作成する。**

バブルごとに、正常処理、例外処理、異常処理に関するミニ仕様を作成する。

㊓　ＤＦＤ作成上の留意点

❶ データフローにはそのデータを特徴づけた名称を付ける。
❷ データフロー同士を交差させない。
❸ 制御を示すようなデータフローを記述しない。
❹ 記述する処理はシステムが正常に稼働している場合とする。
❺ 各バブルには０から連番を階層的に付ける。
❻ ファイルの記述はアクセスを伴うバブルが生じた時点に記述する。

㊔　ＤＦＤの階層的表現のルール



＊上位レベルのＤＦ０１～ＤＦ０３のデータフローは下位レベルで必ず現れる。
＊下位レベルだけのＤＦ０５～ＤＦ０７は上位レベルには現れない。
＊下位レベルだけのデータストアＤＳ０１は上位レベルには現れない。

❶ **最初にコンテキストダイヤグラムを作成する。**

コンテキストダイヤグラムは最上位のダイヤグラムであり、対象業務の範囲を明確にするのに用いる。処理は１つだけで、対象業務全体を１つの機能として表現する。処理の識別番号は「０」にする。入出力データフローは主要なものだけを記入する。

❷ **対象業務機能を詳細化する各レベルのＤＦＤを作成する。**

一面に表現する処理の数は多くとも７つぐらいにする。

❸ **処理の識別番号はその処理のレベルがわかるように付ける。**

それぞれのＤＦＤダイヤグラムに識別番号を付ける。処理番号とダイヤグラム番号は対応づけて相互参照できるようにする。

❹ **機能を下位に展開するとき、入出力データフローに矛盾が発生しないようにする。**

上位機能の入出力データフローやデータストアは、下位に展開したときも、必ず、現れなければならない。下位レベルの中だけに現れるデータフローやデータストアは、上位レベルのＤＦＤには記入する必要はない。

❺ **各機能の下位への展開は、機能ごとに検討する。**

階層の深さは各機能ごとに対応し、全体として合わせる必要はない。上位レベルの２個以上の機能を１つにして詳細な機能に展開しない。

❻ **階層の深さは、せいぜい５レベル程度に収まるようにする。**

レベル数が多くなると、理解しにくくなる。

ⓒ　構造化チャート

㋐　構造化チャートとは

構造化チャートはシステムの構成などを階層的に図式で表現する。サブシステム間、プログラム間、モジュール間などの機能の従属関係を階層的に図式を使用して表す技法である。

㋑　構造化チャートの特徴

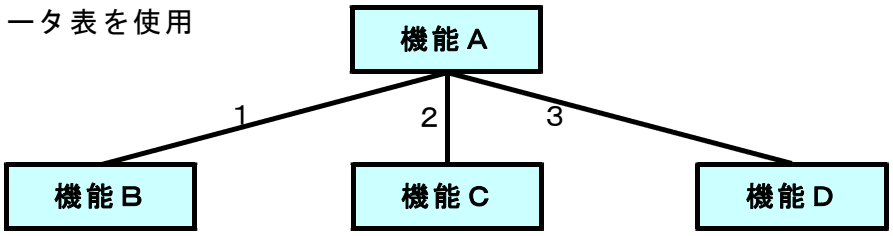
❶ パラメータの記述ができるため、機能（プログラム）間のインタフェースが理解しやすい。

❷ 階層構造で表現するため、システム（サブシステム）を構成しているプログラム構造が理解しやすい。

❸ ループ、判定などの手続きを表現することができる。

㋒　構造図の記号

パラメータ表を使用



番号	入力	出力
1		a
2	a	b
3	b	c

❶ ボックス

機能を表現する記号である。

❷ 連結線

従属関係を表現する記号で、識別のために番号をつけることがある。

❸ パラメータ

機能間のインタフェースを明確にするために、パラメータを記入する。パラメータにはデータ名、データの入出力関係などを明記する。出力は階層構造の下位から上位に向かう方向になる。

❹ パラメータ表

連結線の番号と入力パラメータと出力パラメータを記述する。パラメータが多い場合にパラメータ表を用いると有効である。

ⓓ ＨＩＰＯ

㋐ ＨＩＰＯとは

システムの持つ機能を、階層構造を表す図式目次と、入力、処理、出力を表すＩＰＯダイアグラムで表現する。ＩＰＯ図には、総括ＩＰＯ図と詳細ＩＰＯ図がある。

㋑ ＨＩＰＯの特徴

❶ 外部設計、内部設計、プログラム設計、保守の各工程で使用でき、文書の体系化ができる。

❷ トップダウン設計に適合している。

❸ 機能と入出力に注目し、手順については考えない。

❹ 図式表現で見やすく、わかりやすい。

❺ 設計手法として使用し、設計と同時に文書を作成することができる。

㋒ 図式目次

❶ システム、サブシステム全体の機能を階層構造として表現したものである。

❷ システム全体が、どのような機能から構成されているかを容易に理解できる。

❸ 各機能の詳細を知りたいとき、どのＩＰＯダイアグラムを見ればよいかがわかる目次となる。

❹ 階層構造の最上位にシステムの機能が記述される。

❺ 構造化チャートとの違いは、判定、ループを記述しないことである。

❻ インタフェースに関する記述は、補足説明に記述できる。

㋓ ＩＰＯダイアグラム

図式目次で表示されている各機能について、入力、処理、出力を記述したものである。総括ダイアグラムは図式目次で表された各機能を一つのダイアグラムで記述したものである。詳細ダイアグラムは各機能を個別のダイヤグラムに詳細に記述したものである。入力欄に入

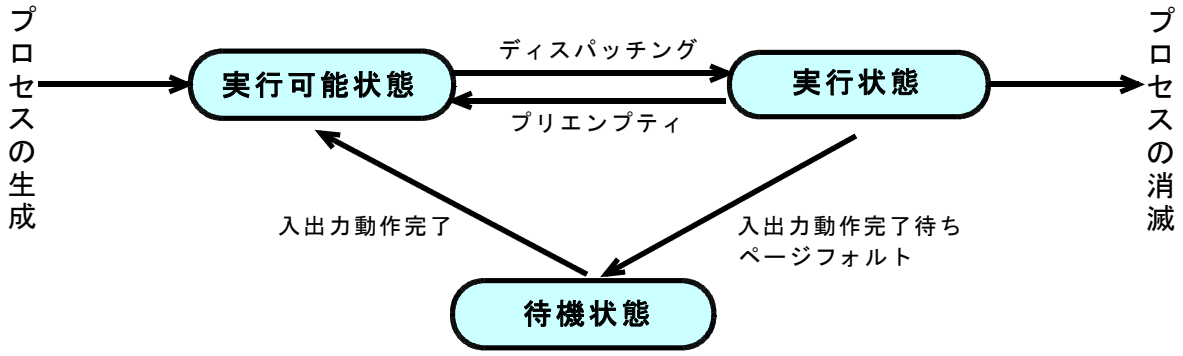
力ファイルを、処理欄に処理内容を、出力欄に出力ファイルを記述する。

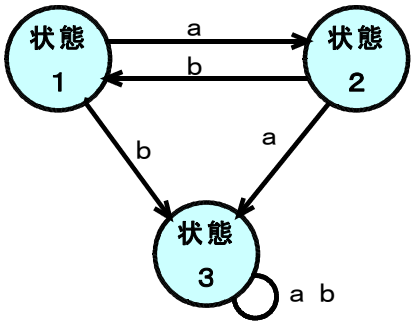
ⓔ　状態遷移図・遷移表

㋐　状態遷移図とは

内部状態の推移関係を図示したもので、オーマトンの状態遷移を表すために考え出されたものである。資源の厳密な管理が必要とされる場合に用いられる。状態遷移図は複雑な処理を精密に描いたり、概念として表現したり、可能な状態の遷移表現に用いる。エンティティやスイッチ、変数、ある与えられた数の状態の間を移り変わっていく複雑な論理を表現する。

㋑　状態遷移図の使用する記号





状態 ＼ 入力	a	b
状態１	状態２	状態３
状態２	状態３	状態１
状態３	状態３	状態３

❶　状態

円で使用し、現在の状態を示す記号で、円の内部に状態を記述する。

❷　状態の遷移

ある状態から別の状態の遷移を矢印で示す。

㋒　遷移表

現在の状態と遷移のきっかけになるイベントとしての入力をマトリックスで表現して、遷移する状態を表したものである。状態遷移図の内容を表形式にまとめたものである。

例題演習

システム開発におけるウォータフォールモデルの説明はどれか。

ア　一度の開発ですべてを作るのではなく，基本的なシステムアーキテクチャの上に機能の優先度に応じて段階的に開発する。
イ　開発工程を設計，実装，テストなどに分け，前の工程が完了してから，その成果物を使って次の工程を行う。
ウ　試作品を作り，利用者の要求をフィードバックして開発を進める。
エ　複雑なソフトウェアを全部最初から作成しようとするのではなく，簡単な部分から分析，設計，実装，テストを繰り返し行い，徐々に拡大していく。

解答解説

ウォータフォールモデルに関する問題である。

ウォータフォールモデルは、基本計画からテストまで、川の流れのように一定方向に作業を進めるのを原則としている開発モデルである。システム開発を基本計画、外部設計、内部設計、プログラム設計、プログラミング、テストの工程順に進め、後戻りせずに開発を進める方式である。開発工程を設計、実装、テストなどに分け、前の工程が完了してから、その成果物を使用して次の工程を行う開発法である。

アはインクリメンタルモデル、イはウォータフォールモデル、ウはプロトタイプモデル、エはスパイラルモデルである。求める答えはイとなる。

例題演習

ウォータフォールモデルによるシステム開発工程の作業内容ａ～ｆを，実施する順序で並べたものはどれか。

〔作業内容〕
ａ　現状の問題点を調査・分析し，対象システムへの要求を定義する。
ｂ　システムとして必要な機能をプログラムに分割し，処理の流れを明確にする。
ｃ　詳細な処理手順を設計し，コーディングする。
ｄ　テストを行う。
ｅ　各プログラム内の構造設計を行う。
ｆ　システムの要求仕様を基に，システムとして必要な機能を定義する。

ア　ａ，ｂ，ｆ，ｃ，ｅ，ｄ　　イ　ａ，ｆ，ｂ，ｅ，ｃ，ｄ
ウ　ａ，ｆ，ｂ，ｅ，ｄ，ｃ　　エ　ａ，ｆ，ｅ，ｂ，ｃ，ｄ

解答解説

ウォータフォールモデルの実施順序に関する問題である。

各工程の内容を整理すると次のようになる。

①　基本計画は現状の問題を洗い出し解決策を検討しシステム基本計画書をまとめる。

② 外部設計は要求仕様をもとに機能を確定しシステム構成を明確にする。
③ 内部設計はシステム構築上必要な機能をプログラムに分割し、プログラム間の処理を明確にする。
④ プログラム設計は内部設計書に基づいて各プログラムの構造設計を行う。プログラムの属性を決定し、モジュールに分解し、モジュール間のインタフェースを決める。
⑤ プログラミングはモジュール内の詳細処理手順を設計・コーディング、テストする。
⑥ テストは結合テストやシステムテストを実施する。

ａは要件定義、ｂは内部設計、ｃはプログラム作成、ｄはテスト、ｅはプログラム設計、ｆは外部設計である。工程の順序は、要件定義（ａ）、外部設計（ｆ）、内部設計（ｂ）、プログラム設計（ｅ）、プログラム作成（ｃ）、テスト（ｄ）となる。ａ、ｆ、ｂ、ｅ、ｃ、ｄとなり、求める答えはイとなる。

例題演習

システム開発の手法の一つであるウォータフォールモデルの説明として、適切なものはどれか。

ア アプリケーションの部分単位に設計・製造を行い、これを次々に繰り返す。
イ システム開発を工程順に進め、後戻りせずに開発を進める。
ウ 動作可能な試作品を作成し、要求仕様の確認・評価を早期に行う。
エ ユーザの参画、少人数による開発、開発ツールの活用によって短期間に開発する。

解答解説

ウォータフォールモデルに関する問題である。

ウォータフォールモデルは、基本計画からテストまで、川の流れのように一定方向に作業を進めるのを原則としている開発モデルである。システム開発を基本計画、外部設計、内部設計、プログラム設計、プログラミング、テストの工程順に進め、後戻りせずに開発を進める方式である。分割された各工程の成果を段階的に確認しながら開発を進める段階的アプローチ法である。上流工程ほど厳しい基準の設定が必要となるが、ユーザの要求分析に曖昧さが残り開発を進める上でその後の各工程に多くの問題を発生させるケースが多い。

ウォータフォールモデルの長所・短所

①目的がはっきりしているときは、その目的に合った効率的なシステム開発ができる。②大規模なシステム開発に向いている。③メンバの役割分担が明確であり、作業に習熟できる。④開発費用や工数が多くなる。⑤開発期間が長期にわたるため、その間にユーザの要求が変化することも多く、それに対してタイムリーに対応できない。⑥抽象的な仕様を段階的に具体化していくという手順を踏むため、最終工程が終わるまで、仕様が１００％実現できるのか判らない場合がある。⑦要求仕様の段階で不具合が入り込みやすく、上流工程で入ったバグはユーザの目に触れる下流工程まで隠れてしまうことが多い。⑧現実のプロジェクトでは反復作業が常に生じるが、工程間のフィードバックが簡単にできない仕組みになっている。

アはスパイラルモデル、イはウォータフォールモデル、ウ、エはプロタイプモデルである。求める答えはイとなる。

例題演習

ソフトウエア開発手法の一つであるプロトタイピングの特徴の記述として，適切なものはどれか。

ア　基本計画，外部設計，内部設計，プログラム設計，プログラミング，テストの順に進めていくので，全体を見通すことができ，スケジュールの決定や資源配分が容易にできる。

イ　システム開発の早い段階で試作品を作成するので，利用部門と開発部門との認識のずれやあいまいさを取り除くことができる。

ウ　ソフトウエアの性質を，仕様が固定的で変更の必要がないものと，仕様の変更が必要であるものとに分類し，仕様の変更があるものについて作成・見直し・変更のプロセスを繰り返す。

エ　大規模アプリケーションを独立性の高い部分に分割し，その部分ごとに設計，プログラミング，テストの工程を繰り返し，徐々にその開発範囲を広げていく。

解答解説

プロトタイプモデルの特徴に関する問題である。

プロトタイプモデルは、ウォータフォールモデルの難点を解決するために考えられたモデルである。ユーザの要求仕様を開発の早い段階に目に見える試作品として現実化する手法である。ユーザの認識結果を開発者にフィードバックし、認識のずれや曖昧さを取り除くことができ最終段階での食い違いをなくすことができる。

プロタイプモデルの特徴は次の通りである。

短期間で開発を完了する。ユーザの意向を反映しながら、システム開発が進められる。小規模なシステムの開発に適している。オンラインシステムやネットワークシステムの開発に適している。開発者が多方面の知識や技術を要求される。

アはウォータフォールモデル、イはプロトタイプモデル、ウ、エはスパイラルモデルである。求める答えはイとなる。

例題演習

プロトタイピングの特徴として、適切なものはどれか。

ア　実際に運用するソフトウェアと同じものをプロトタイプで実現しないと、プロトタイピングの目的は達成できない。

イ　短期間で暫定的に動作するソフトウェアを作り、利用者に試用・評価してもらい、修正を繰り返しながら、仕様を確定していく。

ウ　船などを作る場合、模型を作ることによって製品イメージを明確にするが、模型は必ずしも水に浮く必要はない。ソフトウェアのプロトタイプも同様に、コンピュータ上で動かなくてもよい。

エ　プロトタイピングでは利用者を開発過程に巻き込むことが難しいので、利用者の参加意識の向上を図りたい場合は、プロトタイピングの手法は適用すべきではない。

解答解説

プロトタイプモデルに関する問題である。

プロトタイプモデルは、開発工程の早い段階で目に見える形でユーザが要求を確認できるように、試作品を作成しながらシステムを構築していく手法である。プロトタイプはユーザの要求や意見を明確にするために作成する画面イメージやデザインなどの試作品のことである。

求める答えはイとなる。

例題演習

ソフトウェア開発のプロセスモデルのうち，開発サイクルごとにリスクを最小にしながら，開発サイクルを繰り返すことによって，システムの完成度を高めていくプロセスモデルはどれか。

ア　ウオータフォールモデル　　イ　スパイラルモデル
ウ　成長モデル　　エ　プロトタイピングモデル

解答解説

スパイラルモデルに関する問題である。

アのウォータフォールモデルは、システムの開発工程を、基本計画、外部設計、内部設計、プログラム設計、プログラミング、テストのフェーズに分け、上流から順次作業を進めていく方法で前のフェーズに戻ることがない開発手法である。

イのスパイラルモデルは、ウォータフォールモデルを何度か繰り返して、機能を追加していくインクリメンタルモデルを改良したモデルで、ユーザの要望を取り入れながら、サブシステムごとに開発プロセスモデルを繰り返して進めていく方法で、開発リスクを最小化する。開発単位が独立している場合に適している。求める答えはイとなる。

ウの成長モデルは、ソフトウェアの変更要求の度に、モデリング、要求定義、実装、評価、再モデリングを繰り返す方法である。仕様変更要求に柔軟に対応できるが、プロジェクト管理がやりにくい問題がある。

エのプロトタイピングは、試作品をユーザに見せ、それをたたき台にユーザの要求を聞きながら、システムを完成させていくソフトウェア開発手法である。早い段階でユーザの要求を明確に理解し、開発者との認識のずれをなくすことができる。

例題演習

スパイラルモデルのソフトウェア開発プロセスに関する記述として、正しいものはどれか。

ア　開発コストなどによってリスクを評価しながら開発するので、リスクが最小になる。
イ　基本的に手戻りを許さないので、仕様変更には著しく大きな工数を要する。
ウ　最初からユーザ要求仕様が明白な場合に、最も効率的である。
エ　設計工程で作成したドキュメントをテスト工程で活用することによって、品質を向上できる。

解答解説

スパイラルモデルに関する問題である。

スパイラルモデルはウォータフォールモデルとプロトタイプモデルの両方のよい部分を取り入れた開発手法で、システムの範囲を限定して利用者の要求を聞き取り、仕様をまとめ、プログラミング、テストを行い、利用者の評価と確認をとる。利用者が満足するまで設計、プログラミング、テストを繰り返し、リスクを評価しながら開発するので、リスクが最小になる。

アはスパイラルモデル、イ、ウ、エはウォータフォールモデルである。求める答えはアである。

例題演習

システムの分析・設計に用いる手法でＤＦＤに記述されるものとして、適切なものはどれか。

ア　アルゴリズム　　イ　イベント

ウ　データの流れ　　エ　入出力装置や外部記憶装置

解答解説

ＤＦＤの記述内容に関する問題である。

アのアルゴリズムは、処理手順であり、流れ図に関係する。

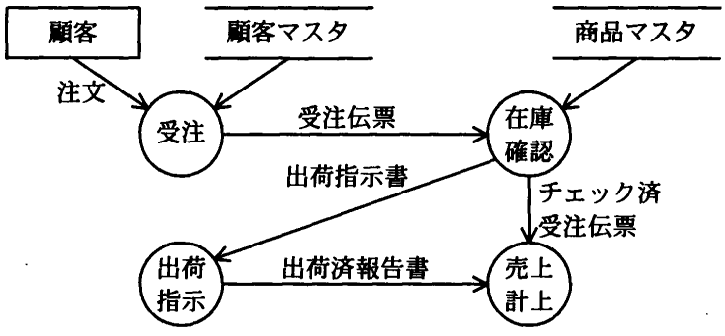
イのイベントは、状態変化を引き起こす出来事で、状態遷移図における遷移のきっかけになる出来事である。

ウのデータの流れは、ＤＦＤにおけるデータ処理を行う機能間の流れである。ＤＦＤに記述されるものはデータの流れで、求める答えはウである。

エの入出力装置や外部記憶装置は、中央処理装置の周辺装置に相当するものである。

例題演習

次の図で用いられている表記法はどれか。



ア　ＤＦＤ　　イ　状態遷移図　　ウ　流れ図　　エ　ペトリネット

解答解説

ＤＦＤに関する問題である。

アのＤＦＤは、システムをデータの流れを中心に表現する手法で、データフロー、処理、デ

ータストア、外部の四つの記号を使用する。求める答えはアである。

イの状態遷移図は、内部状態の推移関係を図示したものである。各種状態をとる中で一時点では１個の状態をとり、ある事象が発生ずると状態は遷移する。

ウの流れ図は、処理手順を図形で表示するもので、処理手順に従って記述されているため処理手順の誤りなどを容易に発見できる、プログラムの経験がなくても使用することが可能なことやファイルやデータの受け渡しが明確になるなどの特徴がある。

エのペトリネットは、事象応答分析に基づいた要求モデルを表現するもので、並列的に動作する機能間同士の同期を表現することが可能なために、制御中心のシステムの要求分析に最適である。ノードと有向矢印、場所・ノードに記入された印によって、作用の推移状況や事象同士の同期のタイミング等を表すことができる。

例題演習

ＤＦＤの表記方法として，適切なものはどれか。

ア　２本の平行線は同期を意味し，名前は付けない。
イ　円には，データを蓄積するファイルの名前を付ける。
ウ　四角には，入力画面や帳票を表す名前を付ける。
エ　矢印には，データを表す名前を付ける。

解答解説

ＤＦＤの表記法に関する問題である。

ＤＦＤは、業務処理におけるデータの流れを図的に表現したもので、データの源泉・吸収は四角、データフローは矢線、データ蓄積は二重線、処理は円の記号を用いる。

アの平行線はデータの蓄積である。

イの円は処理を表す。

ウの四角は、源泉・吸収であり、データの発信元や送信先の顧客などを指す。

エの矢印は、データの流れで、矢線の上にデータ名を付ける。求める答えはエとなる。

例題演習

状態遷移図の説明として，適切なものはどれか。

ア　階層構造の形でプログラムの全体構造を記述する。
イ　時間の経過や状況の変化に基づいて，そのときの動作を記述する。
ウ　システムの機能を概要から詳細へと段階的に記述する。
エ　処理間のデータの流れをデータフロー，処理，データストア及び外部の四つの記号で記述する。

解答解説

状態遷移図に関する問題である。

状態遷移図は状態の遷移を表現した図で、状態を円で、状態の変化と変化させる条件を矢印

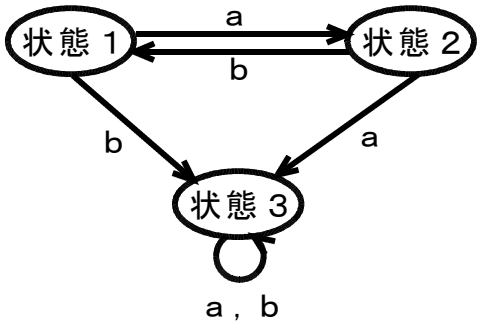
で表現する。タスクの状態遷移やリアルタイム処理の状態変化などを表現するのに利用する。

アは階層構造図、イは状態遷移図、ウは段階的詳細化、エはＤＦＤである。求める答えはイとなる。

例題演習

ａ、ｂを入力とする次の状態遷移図がある。右側の状態遷移表をこの図と等価にするために、表Ａ～Ｄに入れるべき適切な組合せはどれか。

状態 ＼ 入力	a	b
状態１	A	B
状態２	状態３	C
状態３	D	状態３



	A	B	C	D
ア	状態１	状態３	状態２	状態１
イ	状態２	状態３	状態１	状態３
ウ	状態２	状態３	状態２	状態１
エ	状態３	状態１	状態２	状態２

解答解説

状態遷移図、状態遷移表に関する問題である。

Ａは状態２、Ｂは状態３、Ｃは状態１、Ｄは状態３となる。求める答えはイである。

例題演習

ソフトウェアの設計図法の一つであるＨＩＰＯを構成するものの組合せはどれか。

ア　外部，データフロー，詳細ダイアグラム
イ　処理，コンテキストダイアグラム，図式目次
ウ　図式目次，総括ダイアグラム，詳細ダイアグラム
エ　データストア，データフロー，処理

解答解説

ＨＩＰＯに関する問題である。

ＨＩＰＯはシステム設計技法の一つで、システムの機能を３つの構成で階層的に記述する。

図式目次は、システムを構成するモジュールの関係を階層的に表現する。総括ダイヤグラム、詳細ダイヤグラムは各モジュールを入力・処理・出力の形で表現する。図式目次、総括ダイヤグラム、詳細ダイヤグラムがＨＩＰＯを構成する組み合わせで、求める答えはウとなる。

txt040113 オブジェクト指向モデルとＵＭＬ技法

①　クラス、オブジェクト、インスタンス

ⓐ　オブジェクト指向モデル

システムをデータ構造における共通の特性を示すクラスという概念によってモデル化したものである。クラスから生成するオブジェクトは能動的な振舞いを行い、メッセージパッシングが行われると、そのメッセージに対応したオブジェクトが自律的に動作する現象を利用してモデル化を進める。

ⓑ　オブジェクト

オブジェクトは外界における対象物であり、それが具体的なものであれ、概念的なものであれ、ものの存在そのものを示す。オブジェクトは記憶領域のある番地に格納されている符号の実体であり、同じ符号で構成されているオブジェクト同士も、記憶領域内の番地が異なれば別の実体になる。

オブジェクトは存在と状態によってその特性が決定される。車というオブジェクトは、様々な種類、年式、ドアの形状、排気量、乗車定員などの車の状態を表す属性を持っており、これら属性によって車の特性を表すことができる。

オブジェクトは振舞いをもっている。オブジェクトのもつ振舞いを記述したものがメソッドである。車というオブジェクトには、動く、止まる、ドアの開閉などの振舞いがある。

オブジェクトは、プログラムの開始とともにメモリ上に生成され、終了とともに消滅する。

ⓒ　クラス

オブジェクトのうち、共通する性質を持つもの同士をまとめて、新たに名前を付けたものをクラスと呼ぶ。クラスによってオブジェクトを抽象化する。

クラスを構成する共通の性質は、それぞれ属性ごとに定義できる。属性はオブジェクトの内部的な状態を決定するための情報である。一つのクラスは、一つ以上の属性から構成される。同一のクラスに属するオブジェクトは同じ属性をもつ。

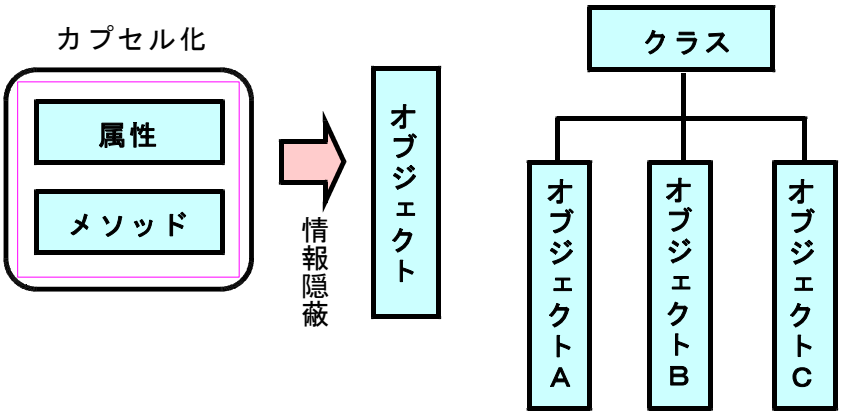
すべてのオブジェクトはどれかのクラスに属しているが、オブジェクトを持たないクラスも存在する。これを抽象クラスという。クラスは静的な符号であり、メモリに常駐する。

ⓓ　インスタンス

クラスに属するオブジェクトを具現化したものである。クラスの属性に具体的なデータが与えられたオブジェクトがインスタンスである。インスタンスはプログラムの実行時に実施される。

②　カプセル化と情報隠蔽

ⓐ　カプセル化



オブジェクトには存在や状態を表す属性と振舞いを表すメソッドが存在する。属性とメソッドを一体にしたものがオブジェクトである。属性とメソッドを一緒にすることをカプセル化と呼ぶ。属性はオブジェクトに格納されており、属性やメソッドはクラスで定義されている。

ⓑ　情報隠蔽

オブジェクトの状態や振舞いをカプセル化して、ブラックボックス化して隠すことを情報隠蔽という。各オブジェクトの属性に対しては、メソッドからしかアクセスできない。オブジェクトの内部に属性は隠され、外部に対してはメソッドだけを公開する。外部からは、属性が隠された状態となり、それをアクセスするためのメソッドだけが見えることになる。カプセル化によって、オブジェクトの独立性が高まり、再利用がし易くなる。

ⓒ　メッセージ

オブジェクトはその内部データを持つとともに、外部からの動作指示によって、その振舞いを実行するという機構を持つ。外部からの動作指示はメッセージによって与えられる。メッセージは、あるオブジェクトに対してその振舞いを動作させるための依頼の単位であり、オブジェクト間のメッセージのやり取りの結果としてメソッドが実行される。

あるオブジェクトがメッセージを送信すると、対応するオブジェクトがそのメッセージを受信して自分の振舞いを実行する。送信したメッセージを受け取ったオブジェクトが、メソッドを起動して自分の振舞いを実行することをメッセージとメソッドの結合と呼ぶ。メッセージはオブジェクトに対して指示できる唯一の手段であり、これ以外の方法でオブジェクトにアクセスさせないことで、オブジェクトの独立性と信頼性を高めることができる。

③　オブジェクトの抽象化

ⓐ　類型化

類型化は構造および操作に関する共通の性質をもつオブジェクトを集め、これを一つのオブジェクトとみなす考え方である。このオブジェクトをクラスという。

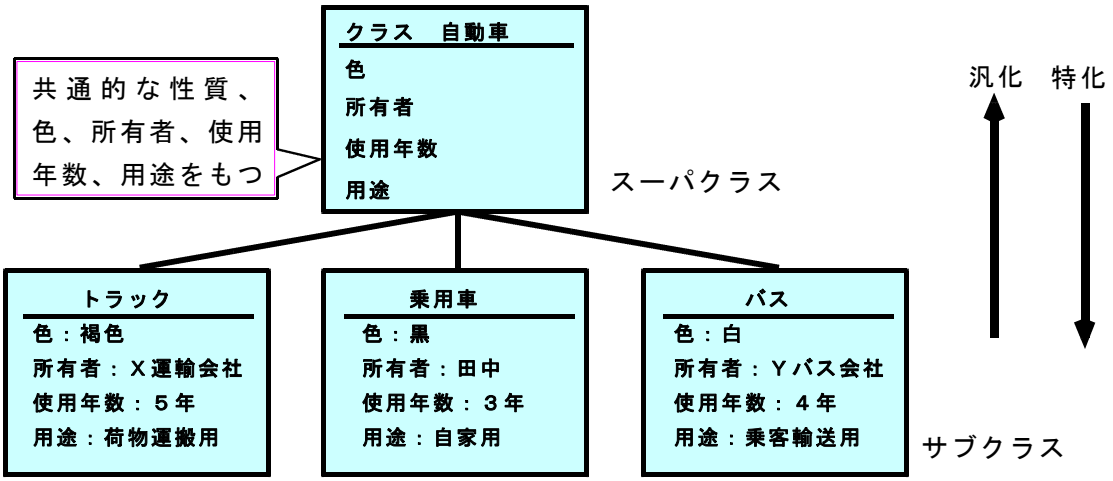
クラスは、オブジェクトのもつ性質を抽出して作られた枠組みであり、その性質は変数あるいはメソッドという形で定義される。オブジェクト指向言語では、はじめにクラスが定義された上で、インスタンスが生成される。クラスの生成されたすべてのインスタンスは、同じ名前と同じ個数のインスタンス変数とインスタンスメソッドをもつ。クラスとそのインスタンスは１レベルの抽象化階層を形成する。これをinstance-of 階層とよぶ。

ⓑ　汎化と特化

汎化は、いくつかのクラスの共通的な性質に着目し、その共通性のみを残し、その他の細かい相違点を無視することによって、より一般化されたクラスを定義する考えである。乗用車、バス、トラックから、汎化によってクラス自動車を定義する。自動車はもとの３つのクラスの共通的な性質、色、所有者、使用年数、用途をもつ。

特化は、あるクラスに対しこれにいくつかの性質を付加することによって、より特殊化されたクラスを定義する考え方である。

汎化と特化の関係にあるクラスを、それぞれスーパクラス、サブクラスといい、これらの間はis-a階層を構成する。is-a階層においてサブクラスのインスタンスは、そのスーパクラスのインスタンスでもある。汎化の考え方によると、スーパクラスの持つ性質はそのサブクラスの性質として通用させることができる。すなわち、スーパクラスの性質はサブクラスに継承されることになる。この継承(インヘリタンス)の概念を適用することによってスーパクラスと同じ性質をサブクラスで繰り返し重複記述する必要がなくなる。



ⓒ　集約化

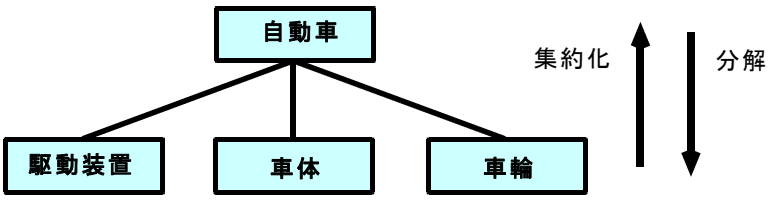
一つのオブジェクトが他のいくつかのオブジェクトから構成されるとき、このオブジェクトは構成要素であるオブジェクトの集約化によって定義される。自動車は、駆動装置、車体、車

輪から構成される場合、自動車というオブジェクトは、三つのオブジェクトである駆動装置、車体、車輪を集約化したものである。集約化の逆は分解である。

集約化によっていくつかのレベルのpart-of 階層が形成される。part-of 階層において、あるオブジェクトの構成要素は、そのオブジェクトの上位のオブジェクトの構成要素でもある。

集約化はあるオブジェクトが存在するためにそのオブジェクトが保持すべき必須の性質として、構成要素であるオブジェクトを記述する。構成要素の組み合わせが異なれば、その集約によって得られるオブジェクトも異なる。駆動装置、車体、車輪の集約化は自動車であり、駆動装置、車体、翼の集約化は飛行機となる。

集約化を利用して定義されたオブジェクトが、その構成要素のオブジェクトのうちの少なくとも１つのオブジェクトを失うと、定義されたオブジェクトも存在しなくなる。



④　継承(インヘリタンス)

ⓐ　継承(インヘリタンス)とは

継承はある性質を引き継ぐことである。オブジェクトの世界では、論理的に同質なものでも、物理的な空間において異質なものとして投影することになる。この投影という考え方が、継承を生み出した。

インヘリタンスの機構はクラス間において存在する。スーパークラスで定義した性質を、サブクラスへ継承することができる。これによって、サブクラスでは同じ性質を定義する必要が無くなる。サブクラスで、新たに必要となった性質は、その都度、サブクラスにおいて定義する。このことを差分プログラミングという。

プログラミングの生産性向上に、継承は非常に有効な手段である。適当なクラスを見つけ、そのクラスの属性やメソッドを利用して新しいクラスを定義することができる。新しく定義したクラスに必要に応じて、差分プログラミングを使用して属性やメソッドを付加することができる。このクラスに属性値を与えることによって、オブジェクトのインスタンスを生成できる。

ⓑ　単一継承と多重継承

サブクラスは複数のスーパクラスから継承することができる。サブクラスから見て、一つのスーパークラスから継承を行うことを単一継承という。複数のスーパークラスから継承を行うことを多重継承と呼ぶ。

ⓒ　クラスライブラリ

オブジェクト指向の一つの目的は開発の生産性の向上である。クラスを極力標準化して、再

利用を図ることによって、生産性の向上が期待できる。クラスライブラリはその目的のために使用される。

クラスライブラリは、定義済みのクラスを提供するライブラリ集で、プログラミングを行う際に、改めてクラスを定義しなくとも、これらの既存のクラスを流用することで開発効率を高めることができる。

⑤　多様化（ポリモルフィズム）

ⓐ　ポリモルフィズム（多様性、多態性）とは

オブジェクト間のメッセージのやり取りにおいて、複数の異なるオブジェクトに対して同一名のメッセージを送信しても、それぞれのオブジェクトが固有の振舞いを実行することができる。これをポリモルフィズム（多様性、多態性）という。マウスのボタンをクリックするだけで種々のメッセージを送信できる。マウスのボタン操作で、画面を上下にスクロールしたり、特定のプログラムを起動したり、ファイルを保存したりすることができる。

ポリモルフィズムは、オブジェクトの自律性を意味しており、同一のメッセージをオブジェクトに送っても、メッセージを受け取ったオブジェクトにより、異なる動作を行う。メッセージを送る側は、受ける側のオブジェクトの構造に関係なく、仕事の依頼のメッセージを送ることができる。メッセージを受ける側に、新しいメソッドが加わったり、メソッドに変更が生じても、受ける側の変更のみで送る側にはなんらの変更も必要としない。送信側および受信側の独立性は保たれることになる。

⑥　オブジェクト指向型開発

ⓐ　オブジェクト指向型開発の特徴

㋐　クラスの再利用

新規プログラミングよりはクラスの再利用に重点を置き、開発の生産性向上を図ることを目標としている。プログラミングを意識する必要が少なくなるので、対象業務を熟知しているユーザの主導のもとで開発することも可能になる。

㋑　スパイラル方式の開発

クラス単位にまとまっているので、次々と並行して詳細クラスの開発を進め、検証を繰り返してシステムを完成させるスパイラル的な開発になる。スパイラルモデル的な開発なので、テストは開発フェースに含まれる。

㋒　高品質なシステムの開発

システムは十分にテストしたクラスの利用により構築されることになり、高品質なシステムが完成する。

ⓑ　オブジェクト指向型開発の効用

㋐　開発の生産性向上

クラスの再利用やインヘリタンス、また差分プログラミングを駆使することにより、従来に比べ開発の手間が大幅に省ける。

㋑　クラスライブラリの構築に伴う品質の向上

クラスの再利用により、システムの品質を向上させることができる。クラスを再利用しているうちにバグがなくなり、そのことによりクラスの品質が向上する。それらのクラスを利用して構築されたシステムの品質が向上することが明らかである

㋒　ユーザ要求に対する柔軟な対応

システムとして完成するかなり前から、クラス単位でプロトタイプをユーザに確認させることができる。オブジェクト指向ではフェーズのギャップが比較的小さいため、従来の方法では厳禁だった工程の後戻りも可能である。これらの特質によって、オブジェクト指向はユーザの要求に柔軟に対応することが可能となる。

㋓　開発コストの削減

開発の生産性向上、工期の削減は開発コストの削減につながる。

⑦　ＵＭＬ

ⓐ　ＵＭＬとは

ＵＭＬはオブジェクト指向モデルを作成するためのモデリング言語である。ＵＭＬを使用することによって、ビジネスのモデル化、開発局面でのソフトウェアのモデル化、更に、静的構造のモデル化、動的振舞いのモデル化など、システム全般のモデル化を可能にするツールである。利用範囲は非常に広く、分析・設計・実装など開発フェーズのあらゆる局面で利用できる。

ⓑ　モデリングとは

問題解決を行ったり、システム化を行う場合、対象の業務を整理し理解する必要が生じる。このような場合に、業務全体やシステム全体をビジュアルに分かりやすく表現するのがモデリングである。個々の機能に着目するのではなく、問題の本質部分を中心に的確に表現することによって問題点が明確になり、対策の検討が容易になる。

モデリングとは複雑な実態を抽象化して図式化することである。

ⓒ　ＵＭＬの主な内容

システムをモデル化する場合のシステムの見方の側面を表すビュー、ビューの内容を表現する図であるダイヤグラム、ダイヤグラムで使用するモデル要素およびモデル要素の使用法で構成される。

ⓓ　ビュー

ビューはモデル化されたシステムのいろいろな側面を表すもので、多くのダイヤグラムからなる抽象構造である。システムの全体像を描写するためには、システムの個々の側面をビューで説明し、そのビューをシステムの全側面について定義する必要がある。どのようなビューを、どれだけ展開すれば、システム全体をモデル化して理解できるかの標準化が示されている。

ビューとして、ユースケースビュー、論理ビュー、コンポーネントビュー、並行性ビュー、配置ビューが使用される。ビューはモデリング言語と開発方法論やプロセスを結びつける役割を果たしている。

ⓔ　ダイヤグラム

㋐　ダイヤグラムはビューの内容を説明するグラフである。

㋑　ＵＭＬには１３種類のダイヤグラムがあり、それらを組み合わせて使用することによって、システムのビューをすべて構成することができる。

㋒　ダイヤグラムは構造図と振る舞い図に分類できる。

㋓　構造図

クラス図、オブジェクト図、パッケージ図、コンポジット構造図、コンポーネント図、配置図などがある。

㋔　振る舞い図

ユースケース図、アクティビティ図、状態図、シーケンス図、コミュニケーション図、相互作用図、タイミング図などが使用される。

ⓕ　モデル要素

㋐　モデル要素は、クラス、オブジェクト、メッセージ、モデル要素間の関係などのオブジェクト指向の概念を表現する。

㋑　モデル要素間の関係には、関連、依存、汎化、集約などがある。

㋒　関連は、クラス間の構造を定義する要素である。

㋓　依存は、ある要素の変更が他の要素に影響を与える関係を表す。

㋔　汎化は具体的なクラスと抽象化したクラスの間の関係を表し、集約はクラス間の全体と部分の関係を表現する。

㋕　モデル要素はいろいろなダイヤグラムで使用されるが、その意味とシンボルはダイヤグラ

ムによらず常に同じである。

ⓖ　構造図の種類

図の名称	定義内容
クラス図	クラスの属性や操作の仕様、クラス間の関連、汎化の関係などを定義する図である。
オブジェクト図	インスタンスとその関係を定義する図である。
パッケージ図	モデルの要素を分類し、複数のパッケージの関係を表現し、定義する図である。
コンポジット構造図	クラスの内部構造を定義する図である。
コンポーネント図	ソフトウェアコンポーネントの構成を定義する図である。
配置図	システムのハードウェア構成を定義する図である。

ⓗ　振る舞い図の種類

図の名称	定義内容
ユースケース図	外部から見た、システムが提供する機能を定義する図である。
アクティビティ図	処理の流れやアクションの実行順序を定義する図である。
状態マシン図	時間の経過とともに変化するオブジェクトの状態を、状態と状態遷移の形で定義した図である。
シーケンス図	ライフライン同士のメッセージのやり取りを時間順に上から下に表現し、クラス間の相互作用を時系列的に定義する図である。
コミュニケーション図	ライフライン同士のメッセージのやり取りをオブジェクトを中心に表現し、クラス間の相互作用を定義する図である。
相互作用概要図	複数の相互作用の実行順序を定義する図である。
タイミング図	相互作用と状態遷移に関する時間制約を定義する図である。

⑧　ビューの種類

ⓐ　ユースケース・ビュー

㋐　ユースケースビューは、アクターがシステムに要求すること、またはシステムがアクターに提供しなくてはならない機能を記述する。アクターはシステムを利用するユーザや連携する他のシステムである。

㋑　システム開発の初期の段階から利用し、ユーザがシステムを利用して何をしたいかを明確にすることを目的として、主にユーザと設計者の間でシステムに対する要求仕様を検討する

ために利用する。

㋒　ユースケース図、アクティビティ図を使用する。

ⓑ　論理ビュー

㋐　システムが提供する機能を静的側面と動的側面で記述する。

㋑　設計者と開発者のためのビューで、クラス図、オブジェクト図、状態図、シーケンス図、コミュニケーション図(コラボレーション図)、アクティビティ図を使用する。

ⓒ　コンポーネント・ビュー

㋐　ソフトウェア・コンポーネント間の依存関係を記述する。

㋑　ソフトウェアコンポーネントには、プログラムのソースコードやバイナリコード、実行可能プログラムなどがあり、これらの依存関係やコードの構造を記述する。

㋒　開発者のためのビューで、コンポーネント図を使用する。

ⓓ　並行性ビュー

㋐　平衡性ビューは並列処理システムの通信や同期・非同期処理の問題を記述する。

㋑　プロセスやスレッドの並列処理における同期処理、非同期処理を記述する。ＣＰＵの資源を有効に使用することや並行して実行しているプロセスやスレッド間の通信の同期を取り扱うことを目的にする。

㋒　主に設計者、システムインテグレータのためのもので、状態図、シーケンス図、コミュニケーション図、アクティビティ図、コンポーネント図、配置図を使用する。

ⓔ　配置ビュー

物理的な装置の配置を記述する。主に設計者、システムインテグレータのためのもので、配置図を使用する。

⑨　各種ダイヤグラム

ⓐ　クラス図

㋐　クラス図はシステム内に存在し、システムを構成するクラス同士の静的な構造を記述する。

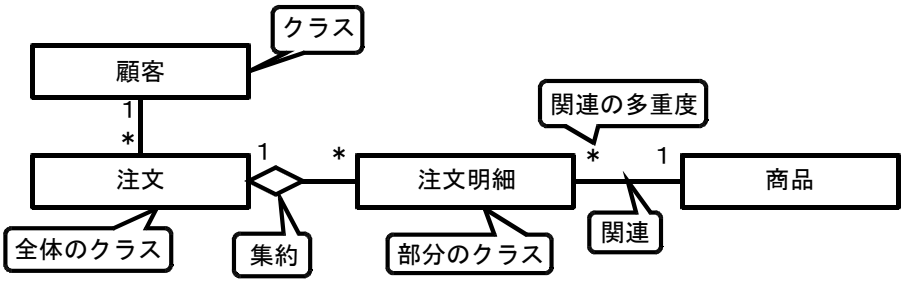
パッケージ単位の表現、システム全体での表現、機能単位での表現など様々な視点で作成することができる。

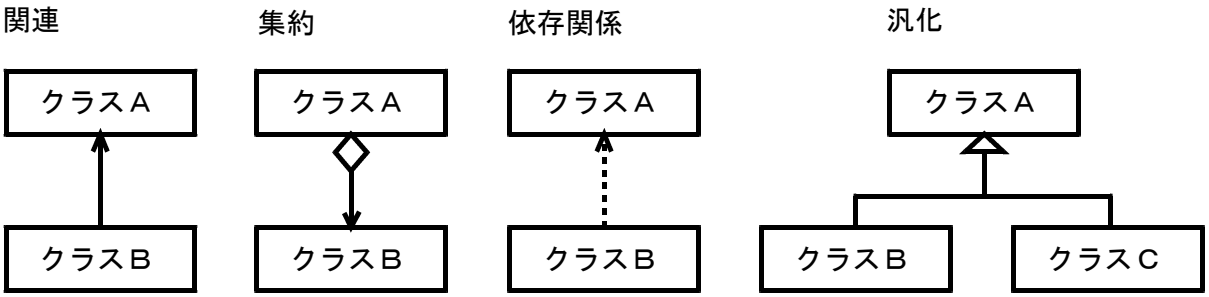
㋑　構成する要素には、汎化、集約、依存関係、多重度など各種の表現が可能である。

多重度は１つのオブジェクトに対応できるオブジェクトの個数を表す要素である。

㋒　クラス図はシステム開発の上流工程から下流工程まで様々なフェーズで使用される。

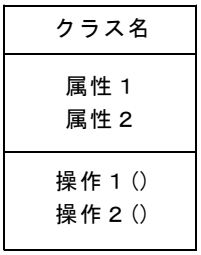
上流工程ではシステム化対象の概念的な構造を整理し、下流工程ではシステムの振る舞いに関係するクラス構造を設計し、システムの再利用性や仕様変更の容易性を高める。



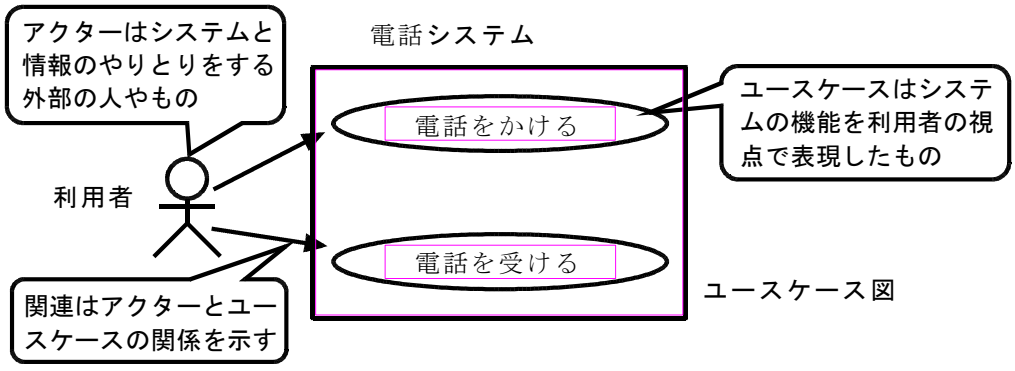


ⓑ　クラスの表記法

クラスは、図に示すように長方形の中にクラス名、属性、操作が記述される。クラス間の関連は、実線または矢印の実線で表す。



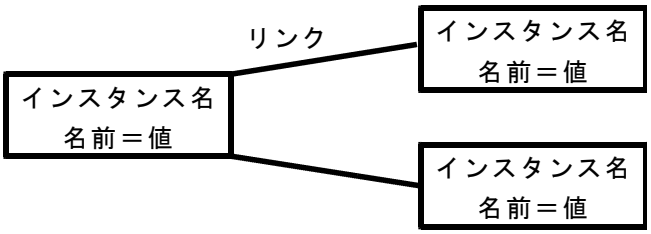
ⓒ　ユースケース図



㋐　ユースケース図は、ユーザのシステムに対する要求を表現したものである。アクターとユースケースを結びつけた図で、両者の関連を示している。ユースケース、アクター、ノートの要素で構成される。

㋑　アクターは、実際にシステムを使用するエンドユーザや接続する他のシステムで、アクターがシステムにメッセージを送信したり、システムからメッセージを受信したりする。

ⓓ　オブジェクト図



クラスから生成される個々のオブジェクト同士の構造を記述する。オブジェクトおよびリンクで構成される。

具体的なものを挙げて、その間の関連性を記述するためにオブジェクト図は使用する。

ⓔ　シーケンス図

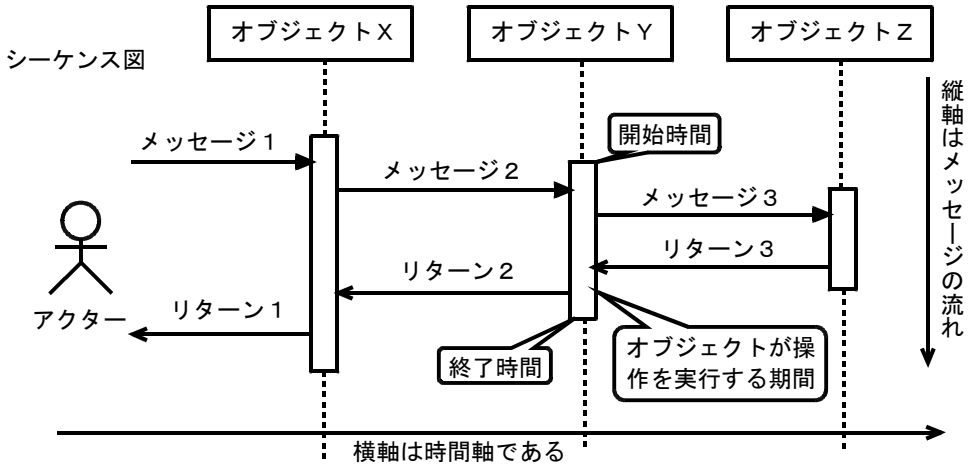
㋐　シーケンス図は相互作用図の一種であり、複数のオブジェクト同士の協調関係を時系列的に表わす図で、システムの動的な側面やオブジェクト間の相互関係を記述する。

㋑　一連の作業する中で協調動作するオブジェクト同士がどのようなメッセージを通信しているかを時系列的に記述する。メッセージ送信の順番が明確な場合に使用する。

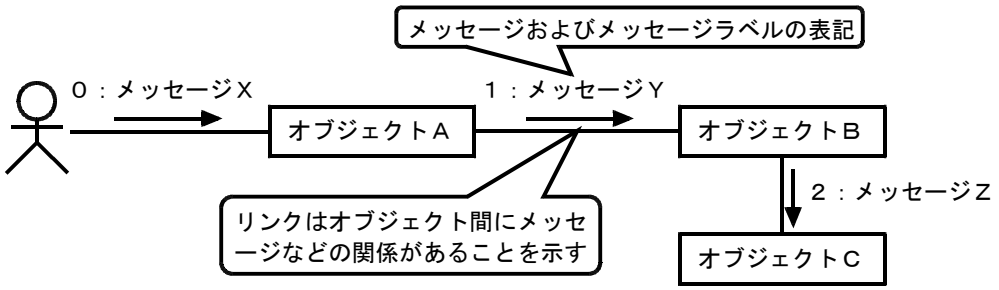
㋒　横軸の時系列に沿って、オブジェクトを複数並べて、縦軸は時間の経過を示し、上から下に向かうメッセージの流れを示す。

㋓　各オブジェクトは四角い箱の中にオブジェクト名を記入し、その箱の下からライフラインという点線を引く。

㋔　構成要素に、オブジェクト、ライフライン、メッセージ、リターン、活性区間などがある。活性区間はオブジェクトが操作を実行している期間を表す。



ⓕ　コミュニケーション図



㋐　複数のオブジェクト同士の協調関係を時系列的に表す図で、システムの動的な側面を記述する相互作用図の一種である。

㋑　シーケンス図が時系列的にメッセージのやり取りを表現するのに対して、コミュニケーション図はオブジェクト間のメッセージのやり取りを、オブジェクトを中心に、接続関係に着目して表現したものである。

㋒　コミュニケーション図の構成要素にはオブジェクト、リンク、メッセージがある。メッセージを送受信するオブジェクト同士を、リンクと呼ばれる直線で結ぶ。リンクしたオブジェクト同士のメッセージには、メッセージ番号とメッセージラベルを付ける。メッセージ番号は、メッセージの実行順序を表す。

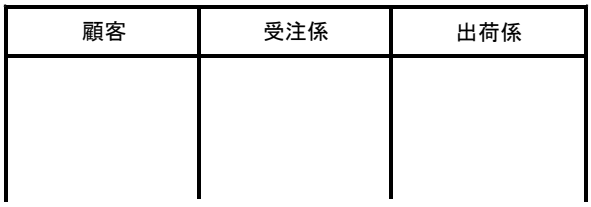
ⓖ　アクティビティ図

㋐　オブジェクトの活動の流れを表す図である。手続きまたはワークフローを記述する。

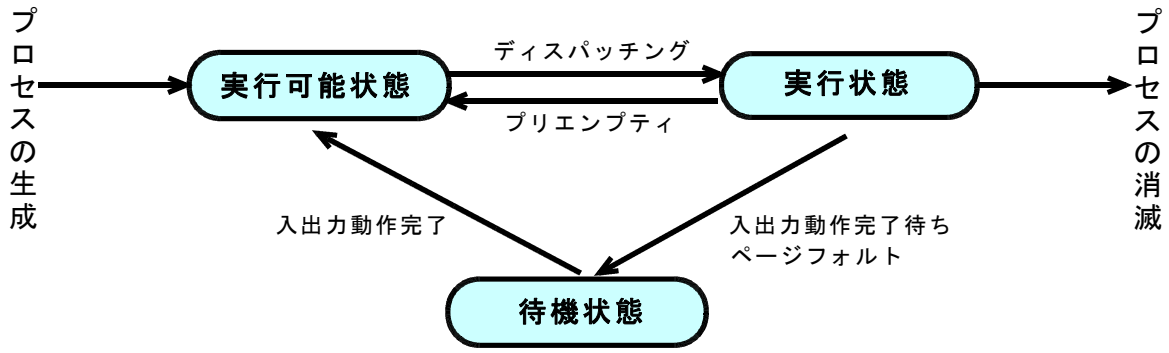
㋑　業務の流れやデータの流れのモデリングに使用する。実行順序を示したり、操作手順やイベントフローを表すのに用いる。

㋒　アクティビティ図は、上流工程から下流工程までさまざまなフェーズで使用する。上流工程ではシステム化対象業務のビジネスプロセスの表現に、下流工程ではフローチャートと同様に内部処理の手順の表現に使用する。アクションを実行する人や組織などを表すアクティビティパーティションと組み合わせて処理の流れを表すのに使用する。

アクティビティパーティション



ⓗ　状態図（ステートチャート図）



㋐　オブジェクトはその内部に状態を持ち、状態は時間やイベントに伴って変換する。状態図はこの時間やイベントに伴って起きる状態の変化を表す図である。

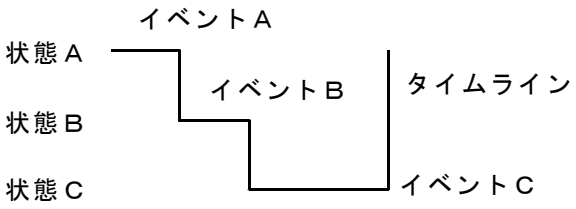
㋑　オブジェクトが生成されてから消滅するまでのライフサイクルの中で、取りうる状態の変化を記述するものである。

㋒　オブジェクトの状態はイベントによって遷移する。イベントは、オブジェクトの属性値や他のオブジェクトとのリンクを変化させる。状態図は、オブジェクトがどんなイベントに反応し、そのイベントによってオブジェクトの内部の状態がどのように変化するかを表している。

㋓　状態は角の丸い長方形で表現し、長方形の中央に状態名を表記する。状態には始点と終点があり、始点は黒丸、終点は黒丸とそれを囲む丸で表現する。イベントはオブジェクトに何らかの影響をもたらす事象の発生である。状態遷移はイベントが発生し、ある状態から他の状態に変わることである。状態遷移は矢線で表し、矢線上にイベント名を表記する。

ⓘ　タイミング図

タイミング図は、相互作用図の一種であり、イベントと状態遷移の関係を表すためにタイムラインを使用して表現する。状態、タイムライン、イベントの要素で構成される。

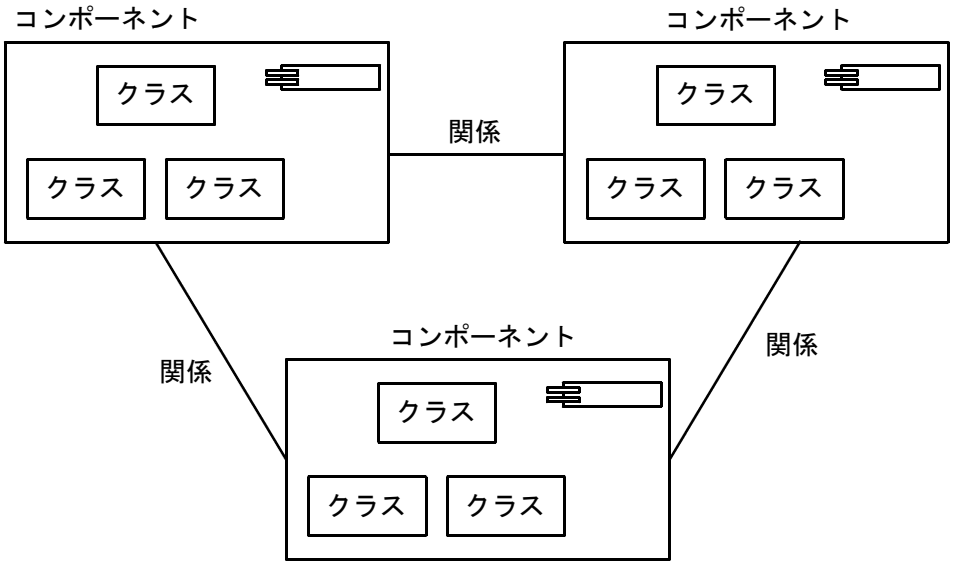


ⓙ　コンポーネント図

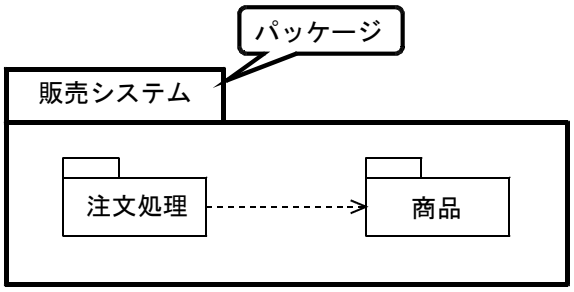
コンポーネント図は、システムの物理的側面であるシステムに使用されるファイルやコンピュータなどの実行環境を表現する実装図の１種である。ソフトウェア・コンポーネントの物理的な構造を記述する。

ソフトウェア・コンポーネントとして、ソースファイル、実行ファイル、ダイナミックリン

クライブラリ、データファイル、データベースの表、ヘルプファイルなどがある。



ⓚ　パッケージ図



パッケージはモデル内の要素を任意のカテゴリ毎にグループ化するための要素である。パッケージを使用してモデル内に存在する様々な要素を整理した図がパッケージ図である。パッケージと依存で構成される。

①　配置図

配置図は、コンピュータや装置の配置とそれらの関係を記述する。

システムを構成するコンピュータ、プリンタ、モニタ、通信関係などのハードウェアの構成や、ソフトウェアコンポーネント、プロセス、インスタンスなどのシステムの実行に必要なコンポーネントの構成を配置図で表現する。

⑩　モデリング

ⓐ　要求モデリング

要求モデリングは、ユーザが開発対象するシステムに対して要求する内容を把握するために

行う。このフェーズではユースケース図などが使用され、システム化対象となるシステムの範囲を明確にする。

ⓑ　分析モデリング

分析モデリングは、要求モデリングを基にシステム化範囲の構造を明確にするために行う。

要求・分析段階では、「どのように実現するか」ではなく、「何を実現するか」に焦点を当てるが、分析段階で重要なのはシステムの概念的かつ論理的な構造と振る舞いを定義することである。このフェーズでは、静的な振る舞いはクラス図等を使用し、動的な振る舞いはコミュニケーション図、シーケンス図、ステートマシン図、アクティビティ図等を使用して記述する。

ⓒ　設計モデリング

設計モデリングは、要求モデリングや分析モデリングを基に、システムの実現方法（どのように実現するか）を定義するために行う。設計モデリングは、抽象的な実装モデリングとしての役割も果たす。このフェーズでは、静的な振る舞いはクラス図等を使用し、動的な振る舞いはコミュニケーション図、シーケンス図、ステートマシン図、アクティビティ図等を使用して記述する。

ⓓ　実装モデリング

実装モデリングは、実際に動作するシステムの構成要素を定義するために行う。モデル駆動型の開発プロセスでは、実装よりも先にモデルを記述するが、既存システムの変更の場合は、ソースコードを利用してモデリングする場合もある。このフェーズでは、コンポーネント図や配置図を使用する。コンポーネント図ではソフトウェア部品の構成を記述し、配置図ではハードウェア構成など、システムの物理的な側面を記述する。プログラムコードと同レベルのクラス図やシーケンス図などを記述することもある。

例題演習

オブジェクト指向の基本概念の組み合わせとして、適切なものはどれか。

ア　仮想化、構造化、投影、クラス
イ　具体化、構造化、連続、クラス
ウ　正規化、カプセル化、分割、クラス
エ　抽象化、カプセル化、継承、クラス

解答解説

オブジェクト指向の基本概念に関する問題である。

抽象化は、いくつかのクラスの共通的な性質に着目し、その共通性のみを残し、その他の細かい相違点を無視することによってより一般化したクラスを定義する汎化のような考え方である。抽象化の反対は具現化である。カプセル化は、データと手続きを一体化し、オブジェクト

が持っている情報を外部から隠し、決められた方法でしかアクセスできないようにすることである。これによって、個々のオブジェクトの独立性を高め、再利用や保守も容易になる。継承は、既存のクラスの機能や性質を受け継いだ新しいクラスを作成することである。クラスは同じ性質を持つオブジェクトの集まりである。

アは、仮想化、構造化、投影が異なる。

イは、構造化、連続が異なる。

ウは、正規化、分割が異なる。

エの抽象化、カプセル化、継承、クラスが基本概念を示す。求める答えはエとなる。

例題演習

オブジェクト指向技術に関する記述として、適切なものはどれか。

ア　インスタンス変数の中には、クラス全体で共有されるデータが格納されている。

イ　オブジェクトは、クラスによって定義される。クラスの中にはメソッドと呼ばれる共有データが格納されている。

ウ　下位クラスはその上位クラスがもつ性質を継承することができるが、複数の上位クラスの性質は継承できない。

エ　メッセージ通信によってオブジェクトに処理を依頼する。それ以外の方法ではオブジェクトに処理を依頼することはできない。

解答解説

オブジェクト指向技術に関する問題である。

アのクラス全体で共有されるデータはクラス変数である。インスタンス変数はオブジェクトの性質を表すもので、同じクラスに属するオブジェクトは共通のインスタンス変数を持つ。

イのクラスの中のメソッドは手続きであって、共有データではない。

ウの複数の上位クラスからの性質の継承は多重継承である。

エのオブジェクトが他のオブジェクトに処理を依頼する場合、メッセージ通信によってのみ可能である。求める答えはエとなる。

例題演習

オブジェクト指向の概念で、上位のクラスのデータやメソッドを下位のクラスで再利用できる性質を何というか。

ア　継承　　イ　カプセル化　　ウ　抽象化　　エ　多様性

解答解説

オブジェクト指向の継承に関する問題である。

アの継承は既存のクラスの機能や性質を受け継いだ新しいクラスを作成することである。求める答えはアとなる。

イのカプセル化はデータと手続きを一体化し、オブジェクトが持っている情報を外部から隠

し、決められた方法でしかアクセスできないようにすることである。

ウの抽象化はいくつかのクラスの共通的な性質に着目し、その共通性のみを残し、その他の細かい相違点を無視することにより一般化したクラスを定義する考え方である。

エの多様性はオブジェクト間のメッセージのやり取りにおいて、複数のオブジェクトに対して同一名のメッセージを送信しても、それぞれのオブジェクトが固有の振る舞いを実行することである。

例題演習

スーパクラスとサブクラスの関係にあるものはどれか。

ア　“会社”と“社員”　　イ　“自動車”と“エンジン”

ウ　“図形”と“三角形”　　エ　“データベース”と“ウィンドウ”

解答解説

オブジェクト指向の汎化・特化に関する問題である。

同じ種類のオブジェクトをまとめたものがクラスである。

アの会社と社員は所有の関係である。

イの自動車とエンジンの関係は集約関係である。

ウの図形と三角形はスーパクラスが図形であり、サブクラスが三角形である。スーパクラスとサブクラスの汎化の関係で、求める答えはウとなる。

エのデータベースとウィンドウは特に関係というものが存在しない。

例題演習

オブジェクト指向における情報隠ぺいの説明として、適切なものはどれか。

ア　オブジェクトは、そのデータに対して定義されたメソッドによってだけアクセス可能である。

イ　オブジェクトは、抽象データ型の要素でなければならない。

ウ　親クラスに定義されたメソッドは、実行時に子クラスだけに引き継がれる。

エ　同一メッセージでも、実行時には受信クラスに基づきメソッドが束縛される。

解答解説

情報隠蔽に関する問題である。

オブジェクトの状態と振る舞いを一体化することをカプセル化といい、カプセル化することによって状態と振る舞いをブラックボックス化することを情報隠蔽という。情報隠蔽された振る舞いを動作させるのは外部のオブジェクトからのメッセージによる。カプセル化によってオブジェクトの独立性が高まり、再利用がしやすくなる。また、メッセージ以外の手段ではオブジェクトにアクセスできないため、オブジェクトの独立性と信頼性を一層高める。

アは情報隠蔽、イは集約化、ウは継承、エは多様性を表す。求める答えはアとなる。

例題演習

オブジェクト指向におけるカプセル化の説明として、適切なものはどれか。

ア　同じ性質をもつ複数のオブジェクトを抽象化して、整理する概念のこと
イ　クラス間に共通する性質を抽出し、共通情報クラスを作ること
ウ　上位クラスの属性とメソッドを下位クラスが引き継ぐこと
エ　データとそれに関する手続を一つにして、オブジェクトの内部に隠蔽すること

解答解説

カプセル化に関する問題である。

オブジェクトの状態と振る舞いを一体化することをカプセル化といい、カプセル化することによって状態と振る舞いをブラックボックス化することを情報隠蔽という。

アは汎化、イは抽象クラス、ウは継承、エがカプセル化であり、求める答えはエとなる。

例題演習

オブジェクト指向に基づく開発では，オブジェクトの内部構造が変更されても利用者がその影響を受けないようにすることができ，それによってオプジェクトの利用者がオブジェクトの内部構造を知らなくてもよいようにすることができる。これを実現するための概念を表す用語はどれか。

ア　カプセル化　　イ　クラス化　　ウ　構造化　　エ　モジュール化

解答解説

カプセル化に関する問題である。

オブジェクトの状態と振る舞いを一体化することをカプセル化といい、カプセル化することによって状態と振る舞いをブラックボックス化することを情報隠蔽という。情報隠蔽された振る舞いを動作させるのは外部のオブジェクトからのメッセージによる。カプセル化によってオブジェクトの独立性が高まり、再利用がしやすくなる。また、メッセージ以外の手段ではオブジェクトにアクセスできないため、オブジェクトの独立性と信頼性を一層高める。

アのカプセル化は、データと手続きを一体化し、オブジェクトが持っている情報を外部から隠し、決められた方法でしかアクセスできない構造にすることである。求める答えはアとなる。

イのクラス化は、オブジェクトに共通するデータ属性とメソッドをカプセル化して、クラスを定義することである。

ウの構造化は、サブシステム、プログラム、モジュールの機能を定義し、その機能間の従属関係を階層的に求める技法である。トップダウンアプローチの考え方を用いる。各機能を定義し、機能間のインターフェースを明確にして、図式化する。

エのモジュール化は、プログラムを特定の機能を持ったいくつかのモジュールに分割することである。分割されたモジュール間にはインターフェースを定義して、階層構造化する。

例題演習

オブジェクト指向で用いられるＵＭＬ(Unified Modeling Language) の説明として、適切なものはどれか。

ア　オブジェクト指向データベースを操作するための言語
イ　オブジェクト指向プログラム言語
ウ　オブジェクト指向分析・設計で、モデルを記述するための表記法
エ　再利用可能なソフトウェア部品の集まり

解答解説

オブジェクト指向のＵＭＬに関する問題である。

ＵＭＬはオブジェクト指向分析で用いられるモデリング言語で、クラス図、オブジェクト図、状態チャート図、ユースケース図などがある。

アのオブジェクト指向データベースを操作するための言語はＳＱＬ３がある。

イのオブジェクト指向言語としては、Ｊａｖａ やＣ＋＋などの言語がある。

ウのモデルを記述するための言語はＵＭＬの内容を表わしている。求める答えはウとなる。

エはクラス・ライブラリである。

例題演習

ＵＭＬのクラス図に記述するものはどれか。

ア　アクティベーション，オブジェクト，ライフライン
イ　オブジェクト，メッセージフロー，リンク
ウ　初期状態，終了状態，遷移
エ　操作，属性，ロール名

解答解説

ＵＭＬのクラス図に関する問題である。

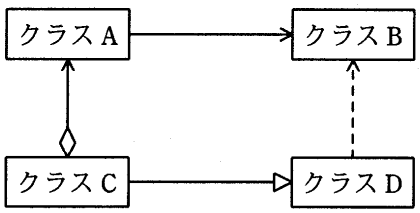
ＵＭＬはオブジェクト指向モデルを作成するためのモデリング言語である。ＵＭＬを使用することによって、ビジネスのモデル化、開発局面でのソフトウェアのモデル化、更に、静的構造のモデル化、動的振舞いのモデル化など、システム全般のモデル化を可能にするツールである。クラス図はシステム内に存在し、システムを構成するクラス同士の静的な構造を記述する。パッケージ単位の表現、システム全体での表現、機能単位での表現など様々な視点で作成することができる。クラスは長方形の中にクラス名、属性、操作が記述される。構成する要素には、汎化、集約、依存関係、多重度など各種の表現が可能である。

アはシーケンス図、イはコラボレーション図、ウはステートチャート(状態図)、エはクラス図である。求める答はエとなる。

例題演習

ＵＭＬのクラス図において，集約の関係にあるクラスはどれか。

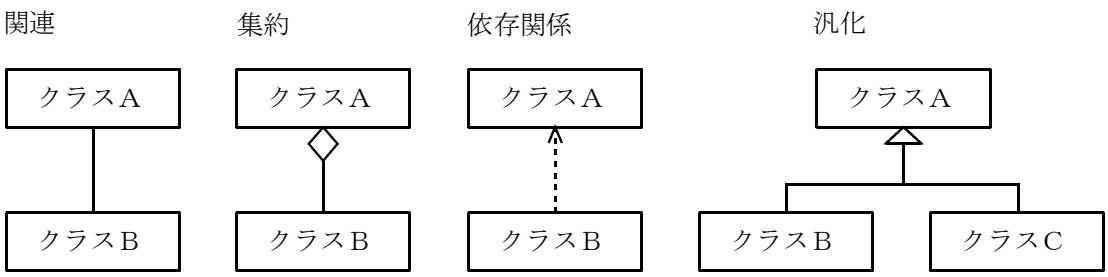
ア　クラスＡとクラスＢ
イ　クラスＡとクラスＣ
ウ　クラスＢとクラスＤ
エ　クラスＣとクラスＤ



解答解説

ＵＭＬに関する問題である。

クラス図は、システム内に存在するクラス同士の静的な構造を記述する。パッケージ単位の表現、システム全体での表現、機能単位での表現など様々な視点で作成することができる。クラスは長方形の中にクラス名、属性、操作が記述される。構成する要素には、汎化、集約、依存関係、多重度など各種の表現が可能である。関連、集約、依存関係、汎化の図的表現法を示すと図のようになる。

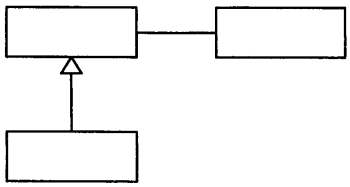


集約の関係を表すのは、クラスＡとクラスＣである。求める答えはイとなる。

例題演習

ＵＭＬにおける図の［　　］の中に記述するものはどれか。

ア　関連名
イ　クラス名
ウ　集約名
エ　ユースケース名



解答解説

クラス図に関する問題である。

クラスは、オブジェクト指向プログラミングにおいて、データとその操作手順であるメソッドをまとめたオブジェクトの雛型を定義したもので、これを定義することによって、同種のオブジェクトをまとめて扱うことができるようになる。クラス図は、システム内に存在し、システムを構成するクラス同士の静的な構造を記述したものであり、長方形の中にクラス名、属性、操作などが記述される。

アの関連は、２つの実体間の関係を捉えたものであり、その関連に名前を付けたのが関連名である。

イのクラス名は、クラスに付けた名前で、クラス図の最も簡単な表記法は、長方形の中にクラス名を記入する。求める答えはイとなる。

ウの集約は、オブジェクトが複数のオブジェクトで構成されるとき、そのオブジェクトが上位のオブジェクトの構成要素となる関係である。この集約に名を付けたのが集約名である。

エのユースケースは、外部から見た、システムが提供する機能を定義したもので、それに名前を付けたのがユースケース名である。

例題演習

ＵＭＬのダイアグラムのうち，インスタンス間の関係を表現するものはどれか。

ア　アクティビティ図　　イ　オブジェクト図
ウ　コンポーネント図　　エ　ユースケース図

解答解説

ＵＭＬダイヤグラムに関する問題である。

アのアクティビティ図は、システムや業務の流れを表現する図、振る舞いを表現する図、アクションの実行順序を定義する図である。

イのオブジェクト図はインスタンスとその関係を定義する図である。求める答えはイとなる。

ウのコンポーネント図はコンポーネントを定義する図である。

エのユースケース図はシステムが提供する機能を定義する図である。

例題演習

オブジェクト指向におけるクラスとインスタンスとの関係のうち，適切なものはどれか。

ア　インスタンスはクラスの仕様を定義したものである。
イ　クラスの定義に基づいてインスタンスが生成される。
ウ　一つのインスタンスに対して，複数のクラスが対応する。
エ　一つのクラスに対して，インスタンスはただ一つ存在する。

解答解説

クラスとインスタンスの関係に関する問題である。

クラスはオブジェクトに共通するデータ属性とメソッドをカプセル化して定義したものであり、インスタンスはクラスに具体的なデータが与えて生成したオブジェクトである。

アはクラスがインスタンスの仕様を定義したものである。

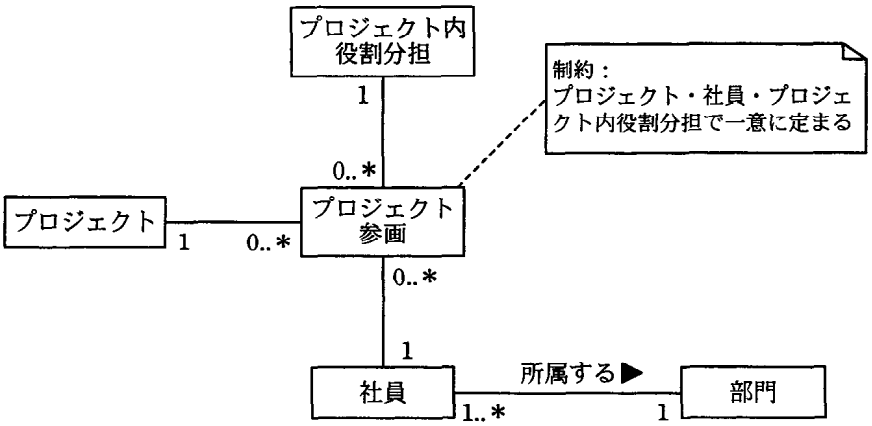
イのクラスの定義に基づいてインスタンスを生成するは適切である。求める答えはイとなる。

ウのインスタンスは一つのクラスから生成される。クラスを生成する場合、複数のクラスから継承して生成される場合がある。

エは一つのクラスに対して複数のインスタンスが生成される。

例題演習

ＵＭＬを用いて表した図のデータモデルに対する多重度の説明のうち，適切なものはどれか。



ア　社員が複数のプロジェクトに参画する場合は，全て同じ役割分担となる。

イ　社員は，同じプロジェクトに異なる役割分担で参画することができる。

ウ　社員は，一つ以上のプロジェクトに参画している。

エ　社員は，複数の部門に所属することができる。

解答解説

ＵＭＬの多重度に関する問題である。

ＵＭＬ、制約条件から考えて、プロジェクト、社員、プロジェクト内役割分担の３条件で一意に定まり、１社員が１プロジェクトに複数の役割分担で参画し、社員とプロジェクト参画は０個以上の１対多の対応関係になることも可能である。

アの複数のプロジェクトに参画する場合は同じ役割分担になるとは言えない。

イの同じ社員が同じプロジェクトに異なる役割分担で参画することができる。求める答えはイとなる。

ウの社員が１つ以上のプロジェクトに参画しているとは言えない。

エの社員は１部門にのみ所属する。複数の部門には所属しない。

例題演習

オブジェクト指向技術を基盤としたソフトウェア部品を組み立てることによってアプリケーションを開発するための技術の総称を何というか。

ア　グループウェア　　イ　コンポーネントウェア

ウ　マクロコマンド　　エ　ミドルウェア

解答解説

コンポーネントウェアに関する問題である。

コンポーネントは構成要素を意味する言葉で、情報処理ではハードウエアやソフトウエアなどの構成要素、あるいは一つの領域や機能などを指す。コンポーネントウェアは、標準仕様に

基づいた再利用可能なソフト部品を組み合わせてアプリケーションを開発する手法またはツールである。ソフト部品はアイコンなどの目に見える形で画面上に表示され、その部品をマウスなどを使って関連付けてアプリケーションを開発する。クラスライブラリやＣＯＭ、ＤＣＯＭ、JavaBeans、ＥＪＢなどが該当する。

アのグループウェアは共同作業の効率化を実現するための支援ツールである。電子メール、電子掲示板、電子会議室、スケジュール管理、共同文書作成支援、データベースなどがある。

イのコンポーネントウェアは、コンポーネントである部品や部品活用のためのツールを利用してシステムを開発する手法で、既存のコンポーネントや新規に作成したコンポーネントを利用してシステムを構築する。オブジェクト指向によるシステム開発はこの考え方を利用している。求める答えはイである。

ウのマクロコマンドは複雑な操作手順や複数のコマンドを１つのコマンドや簡単な操作で実行させたものである。

エのミドルウェアはオペレーティングシステムとアプリケーションソフトウェアの間にあって、利用者にとってよい操作性、豊富なネットワーク機能、使いやすいデータベース、豊富な図形・画像処理機能などを提供するソフトウェアである。

例題演習

オブジェクト指向におけるシステム設計をａ～ｄの工程に分割したとき、最も適切な順序はどれか。

ａ：オブジェクトモデリング
ｂ：カプセル設計
ｃ：業務プロセスモデリング
ｄ：制御設計

ア　ａ→ｂ→ｃ→ｄ　　イ　ａ→ｃ→ｂ→ｄ
ウ　ｃ→ａ→ｂ→ｄ　　エ　ｃ→ａ→ｄ→ｂ

解答解説

オブジェクト指向設計の手順の問題である。

オブジェクト指向設計の手順

① 業務プロセスを検討し、明確化して業務モデルを作成する。

② 業務モデルに基づいて、オブジェクトの識別、オブジェクトのデータ属性とメソッドの識別、オブジェクト間の基本対応の分析、メッセージの分析、オブジェクトモデルの作成の順序でオブジェクトのモデリングを行う。

③ クラスやオブジェクトを作成するカプセル化の業務になる。オブジェクトの必要な属性を設定し、オブジェクトが必要とするメソッドを決定し、属性とメソッドを組み合わせるカプセル設計を行う。

④ オブジェクト間に必要なメッセージやオブジェクトのインタフェースが検討される。必要なオブジェクトが決定されると、これらのオブジェクトを利用して業務を処理する制御

設計を行うことになる。メッセージのやり取りやその間に必要な各種ルールが検討されることになる。

設計順序は、業務モデリング→オブジェクトモデリング→カプセル設計→制御設計の順序になる。ｃ→ａ→ｂ→ｄとなり、求める答えはウとなる。

例題演習

オブジェクト指向プログラミングの特徴はどれか。

ア　オブジェクトが相互にメッセージを送ることによって，協調して動作し，プログラム全体の機能を実現する。

イ　オブジェクトの外部からオブジェクトの内部のデータを直接変更できるので，自由度が高い。

ウ　下位クラスは上位クラスの機能や性質を引き継ぐので，下位クラスに必要な性質は全て上位クラスに含まれる。

エ　個々のオブジェクトが使用するデータ(属性)は，あらかじめデータ辞書に登録しておく。

解答解説

オブジェクト指向プログラミングの特徴に関する問題である。

オブジェクト指向プログラミングの特徴は次のようになる。

① インスタンス機能：クラスから効率的にオブジェクトを生成する。同一のクラスに属するオブジェクトは、共通のメソッドを持つ。

② カプセル化機能：データとメソッドを一体化した機能で、オブジェクト指向プログラミングの生産性と信頼性を向上させる。オブジェクトの独立性が高く、開発時や保守時の波及効果が及ばず、効率のよい作業工程が実現する。

③ メッセージパッシング機能：オブジェクト同士が互いに情報をやり取りし、協調的に問題解決を図るために、メッセージを送受信する機能、実行制御の順序やデータ構造について考量する必要なく、協調的に動作する。

④ インヘリタンス機能：親クラスで指定された性質は、子クラスに引き継がれる。子クラスでは、新たなメソッドを追加することができる。これを差分プログラミングという。対象となる適用業務処理毎に、既存のクラス階層をライブラリとして作成し、再利用が可能になる。

アはメッセージパッシング機能で、協調動作し、プログラムの機能を実現する。求める答えはアとなる。

イは、データとメソッドはカプセル化され、外部から直接操作できない。

ウは、インヘリタンス機能で、下位クラスには差分プログラムによって機能や性質が付加できる。

エは、データはカプセル化され、オブジェクト内部に隠蔽されているためデータ辞書に登録する必要は無い。

txt04021　開発設計

txt040211　ソフトウェアの設計

①　システム要件定義・方式設計

ⓐ　システム要件の定義

次の内容について検討・定義し、評価する。

㋐　システム化の目標と対象範囲
㋑　機能および能力の定義
㋒　業務・組織および利用者の要件
㋓　その他、実行環境要件、周辺インタフェース要件、品質要件

ⓑ　システム最上位レベルの方式確立

㋐　システム方式設計の目的

ハードウェア、ソフトウェア、手作業などに振り分け、必要なシステムの構成を決定する。決定する際には、実現の可能性、リスク、効率的な運用、保守の可能性などを考慮する。

㋑　ハードウェア、ソフトウェア、手作業の機能分割

業務効率、作業負荷、作業コストなどを検討し、決定する。

㋒　ハードウェア方式

信頼性や性能要件に基づいて、冗長化やフォルトトレラント設計、サーバの機能配分、信頼性配分を検討し、ハードウェア構成を決定する。

㋓　ソフトウェア方式

ソフトウェアの自社開発、ソフトウェアパッケージの利用、ミドルウェアの使用などを検討し、ソフトウェア構成を決定する。

㋔　アプリケーション方式

集中処理・分散処理の選択、Ｗｅｂシステム、クライアントサーバシステムなど、システムの処理方式を検討し、決定する。

㋕　データベース方式

データベースの種類を決定する。

②　ソフトウェアの要件定義

ⓐ　ソフトウェア要件の確立

業務モデル、論理データモデルを作成して、システムを構成するソフトウェアに求められる機能、能力、インタフェースなどを決定し、ソフトウェア適格性要件を定める。要件定義のための業務分析には、ＤＦＤ、E-Rダイヤグラム、ＵＭＬなどの分析方法、表現方法を用いる。

この工程で、業務モデリング、データモデリング、インタフェース設計、帳票設計、伝票設計などの作業が実施される。要件確立後、システム要件、システム方式との適合性、実現可能性、保守性などの評価を行う。

ⓑ　業務プロセスと業務モデル

業務プロセスは業務の手順や業務の工程のことである。ユーザニーズの調査で最初に実施するのが業務調査であり、その中で業務の手順や工程を分析し、そこから現状の課題を抽出するのが業務プロセスの調査である。業務分析時に、現状を明示するために業務プロセスフローを作成する。業務プロセスフローの様式はＤＦＤが用いられる。

業務モデルは、業務作業の流れ、業務作業で扱うデータ、業務間の関係などを図解化したものである。システム設計で業務の流れとシステムの関係を明確化するために業務モデルを作成する。この作業を業務モデリングといい、モデリング技法としてはＤＦＤが使用される。

ⓒ　データモデリング

データモデリングは、対象とする業務領域のデータを分析して、データモデルを作成することである。システム設計の作業の一つで、業務で発生する情報を分析し、概念データモデル、論理データモデル、物理データモデルを定義していく一連のプロセスである。

データモデル作成の手順は次のようになる。

㋐　業務モデルで使用されるデータの抽出
㋑　データ項目の操作や更新条件の定義
㋒　主要データのモデル化
㋓　入出力情報の設計

ⓓ　業務分析や要件定義に用いられる手法

㋐　ヒヤリング

ヒヤリングの実施手順を確立する。

㋑　ユースケース

利用者とシステムのやり取りを定義する。

㋒　プロトタイプ

ソフトウェア要求分析において、外部仕様の有効性、仕様の漏れ、実現可能性などの評価を行うために、プロトタイプを作成する。

㋓　ＤＦＤ、Ｅ−Ｒ図、ＵＭＬ

③　ソフトウェアの方式設計

ⓐ　ソフトウェアの方式設計とは

ソフトウェア要件定義書を基に、ソフトウェアの構造とコンポーネントの設計を行う。ソフトウェアをソフトウェアコンポーネント（プログラム）まで分割し、ソフトウェアコンポーネントの機能、ソフトウェアコンポーネント間の処理手順や関係を明確にする。

ⓑ　実行する業務内容

㋐　ソフトウェア構造とコンポーネントの方式設計
㋑　外部およびコンポーネント間のインタフェースの方式設計
㋒　データベース最上位レベルの設計
㋓　利用者文書の作成
㋔　ソフトウェア結合のための要求事項の定義
㋕　ソフトウェア方式設計の評価、レビュー

④　論理データ設計

ⓐ　論理データとは

ユーザの立場からの視点で、複数のデータ項目が集まって意味のある一つの情報あるいはレコードを形成するものである。論理データの設計では、システム内に蓄積するデータ項目やデータ構造を決定する。

ⓑ　論理データ設計の手順

㋐　データ項目の洗い出し

現行業務で使用している台帳や伝票類を参考にして、必要なデータ要素ごとに判りやすい名称を与え、属性や必要桁数を明確にする。洗い出されたデータ項目はバラバラに存在するものである。コードの場合にはコード体系を明確にする。

㋑　データの関連性の分析

伝票や報告書、画面の中で同時に使用されているデータ要素を分析し、同時使用回数の多いデータ要素を一つのグループにまとめる。グループ内の各データ要素の内容を分析し、データ要素間の論理的な関係を検討し、代表するデータ要素をキー項目とする。データ項目が複数個集まって意味のあるレコードが作成される。社員レコード、商品レコードの例を図に示す。

㋒　ファイル候補の決定

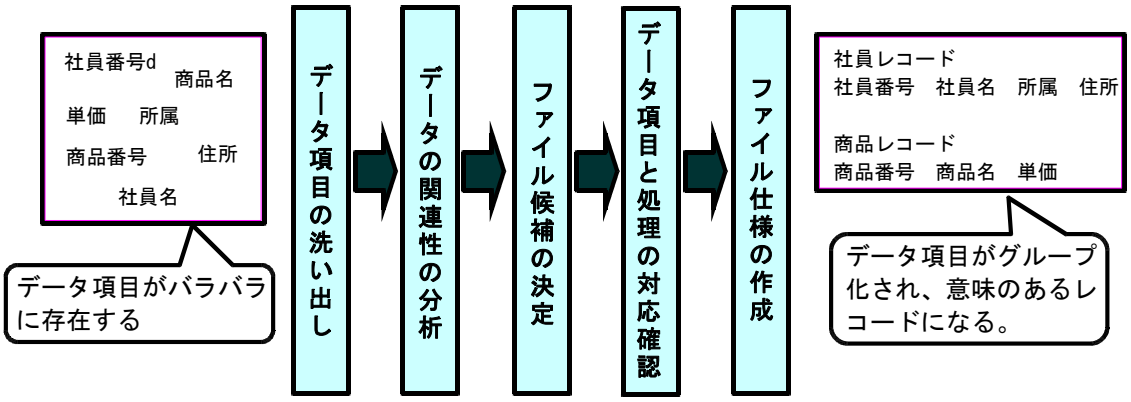
データ項目をグループ化し、一つのファイルにする。

㋓　データ項目と処理の対応確認

データ項目と業務処理を対応づけて、データ項目の過不足を検討する。

㋔　ファイル仕様の作成

ファイルの名称、データ項目名、キー項目、属性、桁数などを決める。



⑤　ソフトウェア方式設計書の作成

ⓐ　ソフトウェア方式設計書

ソフトウェア方式設計の工程で作成した資料に基づいてソフトウェア方式設計書を作成する。ソフトウェア方式設計書はユーザの視点で作成する。ユーザが業務処理を行う時の手順や画面、帳票、データ項目の内容などユーザとのインタフェースの設計が重要である。

ⓑ　ソフトウェア方式設計書の内容

㋐　設計方針

設計方法、文書化技法、設計手順、変更処理などについて記述する。

㋑　システム全体構想

システム化の目的と目標、費用対効果、システム化の範囲などについて記述する。

㋒　適用業務

現状の問題点と解決策、システムの目標と改善点、適用業務処理の概要などについて記述する。

㋓　情報要件・品質要件

入力データ、出力データ、ファイル、データベースなどの要件、データ量に関する発生件数、伸び率、ライフサイクルにおける最大件数、データの属性、レコードの追加・削除の予備件数、信頼性、性能、安全性などについて記述する。

㋔　各種設計仕様

システムのハードウェア、ソフトウェア、ネットワークの構成、仕様について記述する。

㋕　入出力設計書、ファイル設計書、コード設計書

㋖　開発期間、開発要員、開発予算、移行・運用計画

⑥　ソフトウェアの詳細設計

ⓐ　ソフトウェアの詳細設計とは

ソフトウェア詳細設計では、ソフトウェア方式設計書を基に、各ソフトウェアコンポーネントをコーディング、コンパイル、テストを実施するソフトウェアユニット(単体、クラス、モジュール)のレベルに詳細化し、文書化する

プログラムをソフトウェアの構成単位となるコンポーネントとしてとらえ、分割されたコンポーネント(プログラム)ごとの機能およびインタフェースなどについて詳細を決定する。ソフトウェアユニット(単体、モジュール、クラス)への分割、インタフェースとしての入出力、データベース設計、テスト仕様、詳細設計書の作成などの作業を含む。

インタフェース設計では、ソフトウェア要件定義書を基に、操作性、応答性、視認性、ハードウェア、ソフトウェアの機能、処理方法を考慮して、入出力装置を介して取り扱われるデータに関する物理設計を行う。

ⓑ　実行される業務内容

㋐　ソフトウェアコンポーネントの詳細設計

㋑　ソフトウェアインタフェースの詳細設計

㋒　データベースの詳細設計
㋓　利用者文書の更新
㋔　ソフトウェアユニットの要求事項の定義
㋕　ソフトウェア結合のための要求事項の更新
㋖　ソフトウェア詳細設計および要求事項の評価、レビュー

ⓒ　ソフトウェアコンポーネントとは

コンポーネントは組み合わせることのできるソフトウェアの基本単位である。実装を隠蔽してインタフェースの集合を外部に公開し、組み立てることができ、配置することができ、置き換えることができるシステムの部分である。定義されたアーキテクチャ上で明確な機能を果たし、システムの他の要素に関係せずに、独立して容易に取り替えられる。

システムを構成する再利用可能なソフトウェア部品であり、内部のデータ構造や動作を隠蔽し、コンポーネント機能を利用するための外部インタフェースを装備している。

ⓓ　コンポーネントインタフェース

コンポーネント間、または他のシステムと接続する際の、データの形式、データ転送速度、タイミング、プロトコルなどの規格・仕様である。コンポーネント設計工程で、ソフトウェアコンポーネント(プログラム)への分解作業の中で定義される。

インタフェースを明確化することにより、コンポーネントの独立性を高め、コンポーネントの再利用やソフトウェア部品としての流通を図る。

ⓔ　コンポーネント設計

内部設計ともいい、システム設計書に基づいて、ソフトウェアコンポーネント設計、入出力設計、物理データベース設計を行う。

ソフトウェアコンポーネント設計では、内部処理からみた機能への分割、分割されたコンポーネント間のインタフェースなどの設計を行う。入出力設計では操作性や応答性などを考慮して入出力内容、ヒューマンインタフェース仕様などを設計する。物理データベース設計ではファイル設計と論理データベース設計から物理データベースへのマッピングを行う。

⑦　構造化設計

ⓐ　ソフトウェア構造と構造化

ソフトウェア構造はソフトウェアを機能で分割し、段階的に体系化した構造である。システム設計でサブシステム(基本機能)に分割された機能単位を、更にコンポーネント単位に機能を分割して段階的に構造化する。

ソフトウェア設計は、ソフトウェアの要求をプログラムの構造に変換していく作業である。ユーザの要求を段階的に詳細化する過程で、機能単位に分割を進め、構造化を図る。システム

を構成するコンポーネント間やプログラム間の関係が構造化されて分かりやすくなり、開発生産性や保守性が高くなる。

ソフトウェア構造化設計は次の手順で行う

㋐　コンポーネントの分解
㋑　機能仕様決定
㋒　コンポーネント間インタフェースの定義

構造化技法としては、ＤＦＤ、構造化チャートなどが用いられる。

ⓑ　構造化の概念

構造化設計の考え方として、次の概念が利用される。

㋐　分割と統合の概念

大きな規模のソフトウェアの複雑さに対処するため、いくつかの単純な部分に分けて設計し、プログラミングができた時に統合していく考え方である。

㋑　段階的詳細化の概念

基本計画からプログラミングまでのシステム開発工程で、概念設計から詳細設計に段階的に作業を進めていく考え方である。ユーザ向け仕様からシステムの機能まで詳細化したり、プログラムの中で詳細化を図ったり、コンポーネントユニットまで詳細化を進める。

㋒　抽象化の概念

抽象化はソフトウェアの複雑さを軽減するために、ある事実を包括した概念で表現することである。ソフトウェアへのユーザ要求をプログラムへ転換していく時の考え方の一つである。事実を単純化し、複雑な事象を管理しやすくする。

㋓　情報隠蔽の概念

オブジェクト指向でクラスを定義する際に、カプセル化によって内部を隠蔽する。コンポーネントユニットの機能とその機能を使用するためのインタフェースを示すことによって、コンポーネントユニットが使用可能になる。

ⓒ　機能分割・構造化の特徴

㋐　段階的に構造化を行うために設計しやすい。
㋑　構造化の考え方は、外部設計、内部設計、プログラム設計、プログラミングの各段階で使用することができる。
㋒　機能の分割を行う場合、各機能の独立性を考慮しているため、並行して設計やプログラミングを行うことができる。

㊓　分割された機能は、単純で理解し易いため、開発が容易で保守も行いやすい。

ⓓ　機能分割、構造化の手順

㋐　プログラム詳細機能の洗い出し

プログラムの機能を詳細に洗い出し、モジュール分割の準備を行う。作業内容として、図式目次の作成、プロセスフローの作成、総括ダイヤグラムの作成、詳細ダイアグラムの作成を順次行う。

㋑　プログラム間インタフェースの決定

プログラム間で受け渡すデータを明確にする。ＤＦＤなどを利用し、処理される入力データ、出力データの流れを明確にする。データが変換されていく過程や、各プログラム間のインタフェースが明確になる。

㋒　プロセスフローの決定

プログラムの処理順序を決定する。流れ図を利用して、処理の順序、ファイルやデータの受渡を明確にする。

㋓　階層構造化する。

構造化チャートを作成する。

⑧　プロセス設計

ⓐ　プロセス設計とは

ソフトウェア方式設計書を利用して、システムで行う業務の処理手順・処理内容を検討し、使用するプログラムやプログラム間のインタフェースなどを決定し、システムでの処理手順・処理内容をプロセスフロー図にまとめる。プロセスを設計する場合、各フェーズに用いる基本処理パターンを認識して利用する。

ⓑ　プロセス設計に求められる要件

㋐　業務を単位とする一連の処理手順を図式化する。
㋑　入力情報から出力情報を得るまでの処理の過程をフェーズ単位に設定する。
㋒　フェーズごとの入出力ファイルを設定する。
㋓　各ファイルは処理に対応した仕様を満たすように作成する。
㋔　仕様内容として、整列キー、レコードの並び順、データ項目の構成などが問題になる。
㋕　参照したり、更新の対象になるマスタファイルとの関係を検討する。検討内容は、必要な

データ項目、整列のためのキー項目、レコードの並び順、アクセス方式などがある。

㊍　出力帳票の様式、出力媒体の種類、レコード様式について検討する。

㊎　仕様書への展開が可能になるまで、各フェーズにおける処理概要を検討する。

㊏　処理概要では、集計処理、整列処理、照合処理、更新処理、印刷処理などの処理内容が明確になるようにする。

㊐　処理内容に関係する項目についても明記する。例えば、整列処理の整列キー、集計処理のグループ項目、更新処理のデータ項目などがある。

ⓒ　プロセス設計上の留意点

㋐　ハードウェアの機器構成および処理能力を考慮する。

㋑　オペレーティングシステムの機能や提供されるサービスプログラムの内容・用途、使用される関連ファイルなどを周知しておく。

㋒　処理サイクル、入出力のタイミングを考慮する。

㋓　プロセスは可能な限り単純にする。

㋔　インタフェースの最適化を図る。

㋕　データの量、性格、条件を周知しておく。

㋖　エラー条件はできる限りプロセスの入口でつぶしておく。

㋗　例外処理の方法を考慮しておく。

㋘　入力処理、出力処理、ファイル処理、コード設計の各作業と同期を取り、協調しながら作業を進める。

㋙　システムエンジニア自身がプログラミングすることを前提に設計してはならない。

ⓓ　プログラム間のインタフェース

プログラム間のインタフェースの決定は、プログラム間でのデータの受渡を明確にすることであり、データフローを明確にすることである。受渡の手段としてバッチ処理の場合、ファイルが用いられる。

ⓔ　プロセスの構成要素

コンピュータシステムの処理手順を構成する最小の単位をフェーズまたはジョブステップという。フェーズは１本のプログラムと同等の単位である。フェーズが組み立てられて、プロセスが構成される。１つ以上のジョブの集合によって、プロセスが構成される。

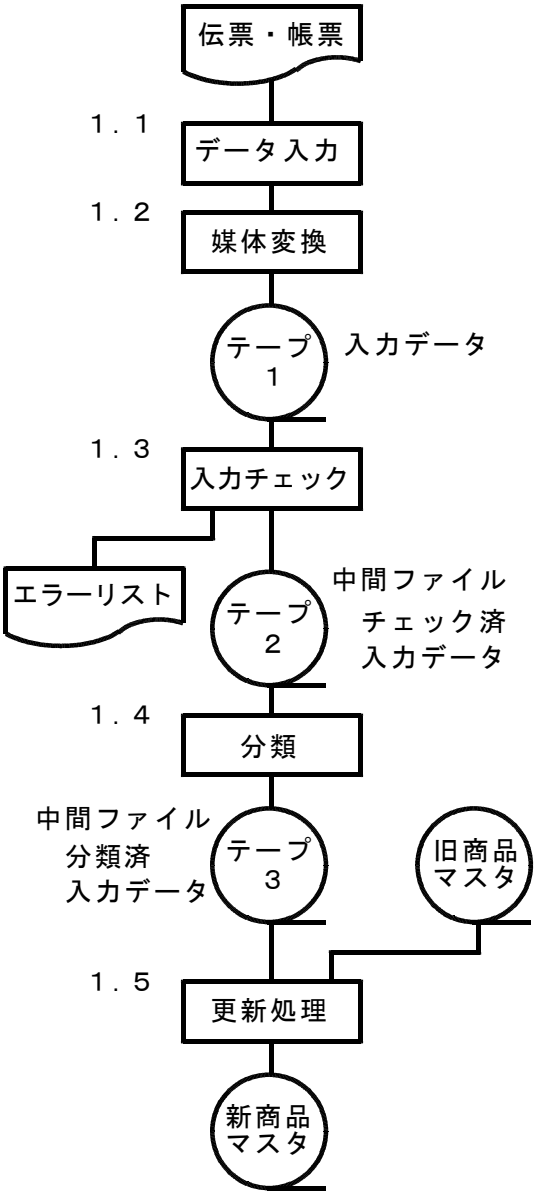
ⓕ　プロセスを構成する基本処理パターン

㋐　入力変換処理

プロセスに最適な媒体に変換する処理。

㋑　併合処理

同一ファイル形式の複数個のファイルを一つにまとめる処理。



処理概要

1．1 & 1．2
＊商品の登録、修正、削除情報を磁気テープに格納する。

1．3
＊入力データの様式チェックを行う。
＊エラーがあるとエラーリストに出力する。
＊正しいデータを中間ファイルに出力する。

1．4
＊分類キー：商品コード
＊商品コードの昇順に並び替える。

1．5
＊中間ファイルの商品マスタ更新データと、商品マスタファイルを突き合わせて、商品マスタファイルを更新する。

㋒　分配処理

入力ファイルの中の特定項目をある条件により比較し、その判定結果により、入力ファイルのデータを別々のファイルに振り分ける処理。

㋓　生成処理

１つ以上のファイルを読み込んで、レコードの変形や加工を行い、入力ファイルとは異なった内容のファイルを作成する処理。

㋔　分類処理

入力ファイルのレコードの記録順序を指定された分類キーの順序に並べ替える処理。

㋕　更新処理

マスタファイルの内容をトランザクションファイルによって、変更・追加・削除し、最新のマスタファイルを作成する処理。

㋖　抽出処理

必要な情報を検索して抜き出す処理。

㋒　照合処理

マスタファイルとトランザクションファイルをマッチングキーにより突合せ、トランザクションファイルの内容を検査したり、マスタファイルの内容を付与する処理。

㋓　複写処理

入力ファイルのレコードを出力ファイルにそっくり書き写す処理。

㋔　集計処理

グループ別に集計する処理である。コントロールブレーク処理のアルゴリズムが用いられる。

㋕　出力変換処理

プリンタ、ディスプレイ、コンソールタイプライタなどの人間が判断可能な媒体に印字または表示する処理。

⑨　物理データ設計

ⓐ　マスタファイルと主要ファイルの項目／編成方法の決定

具体的なファイル媒体や編成方法等を決定する。ファイルの特性の理解、ファイルの編成方法の決定、ファイル媒体の決定、レコードレイアウトの決定、アクセス時間と容量の見積などを順次進める。

ⓑ　レコードレイアウトの設計

ファイル／レコード／フィールドの関係の決定する。レコードタイプ、レコードレイアウトを明確にする。

ⓒ　物理データ設計手順

㋐　ファイル特性の理解

使用率、増加率、機密保護など業務処理上の観点から、ファイルに要求される特性を検討する。

㋑　ファイル編成方針の決定

ファイルの特性、使用するソフトウェアの機能に基づいて、ファイルの編成方式を決定する。順編成、直接編成、相対編成、索引編成、区分編成、仮想記憶編成について、ファイル選択基準に基づいて検討する。格納法、アクセス法、アクセス時間、スペース効率、用途、媒体等を考慮して、編成方式を決定する。

㋒　ファイル媒体の決定

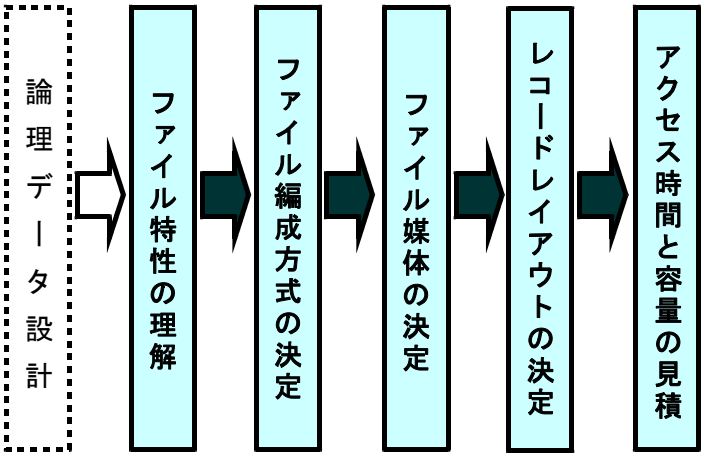
処理上の要求、ファイル編成方式を考慮し、ファイルの媒体を決定する。検討の観点は、処理方式(オンライン処理／バッチ処理)、対象レコードの処理量、要求処理速度、乱処理／順処理などがある。

㋓　レコードレイアウトの決定

レコードを構成する項目とそれらの配列を設計する。将来の項目の追加を考慮し予備のフィールドを設ける。使用装置の制約がある場合、レコードの分割を考える。キー項目のレベルの高いものから配置する。関連性のある項目をまとめる。

㋔　アクセス時間と容量の見積り

アクセス時間は該当レコードを記憶している媒体から主記憶装置に読み込むまでの時間である。ファイルの媒体の種類や編成方式、レコード長やブロック長によって異なる。オンラインリアルタイム処理の場合は、アクセス時間を縮めることが重要である。



⑩　入出力詳細設計

ⓐ　詳細設計作業

原票設計は記入者、コンピュータ処理、保管の３つの観点が重要である。帳票設計は出力イメージ図をもとに、スペーシングチャートに記入する。画面設計は画面イメージをもとに、フィールド属性や色などを決定する。

ⓑ　入力チェック方式

入力データを正しく保つための対策であり、データエラーの種類はデータ自体の誤り、データの過不足、データの遅延等がある。データの修正は発見時に直接修正するより、原票に戻って、記入元でエラーの修正を行うのがよい。プログラムによるエラーの検出は、正しいデータを破壊しないような工夫、あるいは再度同じデータを入力することでデータの重複が発生しな

いような工夫が必要である。

ⓒ　メッセージ設計

システムで用いるメッセージ内容を決定する。コンピュータが実行中に発生したエラーについて、状況説明、エラーの原因、次の操作を指摘する配慮が必要である。利用者へその原因と対応法を画面やプリンタ出力で表示する。利用者にとってわかりやすい、使いやすいメッセージを表示することが重要である。

ⓓ　メニュー設計

メニューは利用者がコマンドを覚えなくても済むように作成することが重要である。熟練者はコマンドを好んで用いるため、メニュー方式を好まない場合がある。初心者や一般の利用者はコマンドに馴染まない傾向があるため、メニュー方式を使用する。従って、コマンドとの関連をメニューに持たせないことが重要である。

⑪　ソフトウェアテストの設計

ⓐ　ソフトウェアユニットテストの設計

ソフトウェア詳細設計書で提示された要件をすべて満たしているかどうかを確認するために、テストの範囲、テスト計画、テスト方式を定義し、ソフトウェアユニットのテスト仕様書を作成する。

ⓑ　ソフトウェア結合テストの設計

ソフトウェア方式設計書で提示された要件をすべて満たしているかどうかを確認するために、テストの範囲、テスト計画、テスト方式を定義し、ソフトウェア結合テスト仕様書を作成する。

⑫　ソフトウェア詳細設計書の作成

ⓐ　ソフトウェア詳細設計書

ソフトウェア方式設計書に基づいて、コンピュータシステムを中心に設計した内容を文書化したもので、システム全体を説明した部分とプログラム単位に記述する部分に分かれる。プログラム仕様書はプログラム単位に記述する。

ⓑ　ソフトウェア詳細設計書の記述内容

㋐　ソフトウェア詳細設計方針

採用した設計技法、文書化技法、設計手順、特記事項について記述する。

㋑　システム構成

システム概要、システム構成、プログラム間インタフェース、プログラム関連図、プログラム一覧などについて記述する。サブシステム構造図、ＤＦＤ、流れ図、プロセスフロー図などを活用する。

㋒　入出力設計、ファイル設計

画面レイアウト、帳票レイアウト、レコードレイアウトについて記述する。

㋓　共通テーブル仕様、共通プログラム一覧

㋔　プログラム仕様

例題演習

開発プロセスにおいて，ソフトウェア方式設計で行うべき作業はどれか。

ア　顧客に意見を求めて仕様を決定する。
イ　ソフトウェア品目に対する要件を，最上位レベルの構造を表現する方式で，かつ，ソフトウェアコンポーネントを識別する方式に変換する。
ウ　プログラム１行ごとの処理まで明確になるように詳細化する。
エ　要求内容を図表などの形式でまとめ，段階的に詳細化して分析する。

解答解説

ソフトウェア方式設計に関する問題である。

ソフトウェアの方式設計は、ソフトウェア要件定義書を基に、ソフトウェアの構造とコンポーネントの設計を行う。ソフトウェアをソフトウェアコンポーネントまで分割し、ソフトウェアコンポーネントの機能、ソフトウェアコンポーネント間の処理手順や関係を明確にする。業務内容は、ソフトウェア構造とコンポーネントの方式設計、外部およびコンポーネント間のインタフェースの方式設計、データベース最上位レベルの設計、利用者文書の作成、ソフトウェア結合のための要求事項の定義、ソフトウェア方式設計の評価、レビューである。

イのソフトウェア品目に対する要件を、最上位レベルの構造を表現する方式で、かつ、ソフトウェアコンポーネントを識別する方式に変換する内容が適切である。

アはシステム要件定義、イはソフトウェア方式設計、ウはソフトウェア詳細設計、エはシステム方式設計である。求める答えはイとなる。

例題演習

システム開発の外部設計工程で行う作業として、適切なものはどれか。

ア　物理データ設計　　イ　プログラム構造化設計
ウ　要求定義　　エ　論理データ設計

解答解説

外部設計工程の作業内容に関する問題である。

外部設計の主要作業

① 要求仕様の確認
② サブシステムの定義と展開
③ 画面設計・報告書の設計
④ コード設計
⑤ 論理データ設計
⑥ 外部設計書の作成

アの物理設計は内部設計工程、イのプログラムの機能化設計はプログラム設計工程、ウの要求定義は基本計画工程、論理データ設計は外部設計工程で行われる。求める答えはエとなる。

例題演習

システム開発で行う次の作業のうち、外部設計の工程で行う作業はどれか。

ア 画面・報告書の設計　　イ システム化計画の立案
ウ プログラム構造化設計　　エ モジュールの論理設計

解答解説

外部設計工程で行われる作業内容に関する問題である。

アの画面・報告書の設計は外部設計、イのシステム化計画の立案は基本計画、ウのプログラム構造化設計は内部設計、エのモジュールの論理設計はプログラム設計の各工程で行う。求める答えはアとなる。

例題演習

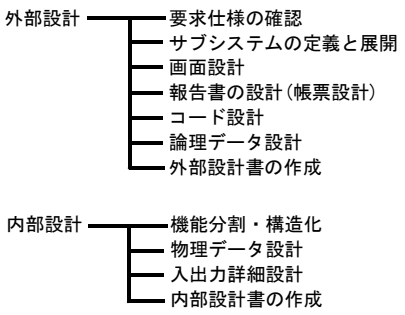
外部設計及び内部設計の説明のうち，適切なものはどれか。

ア 外部設計ではシステムを幾つかのプログラムに分割し，内部設計ではプログラムごとのＤＦＤを作成する。
イ 外部設計ではデータ項目を洗い出して論理データ構造を決定し，内部設計では物理データ構造，データの処理方式やチェック方式などを決定する。
ウ 外部設計と内部設計の遂行順序は，基本計画におけるユーザの要求に基づいて決定される。
エ 外部設計はコンピュータ側から見たシステム設計であり，内部設計はユーザ側から見たシステム設計である。

解答解説

外部設計・内部設計の作業に関する問題である。

右の図に、外部設計、内部設計工程で実施される主要な作業内容を示す。



アのシステムをプログラムに分割するのは内部設計であり、ＤＦＤは両者の工程で使用する。

イの外部設計で論理データ構造、内部設計で物理データ

構造を決定する記述は適切である。求める答えはイとなる。

ウの遂行内容・順序は、外部設計は基本計画のユーザ要求、内部設計は外部設計の結果に基づいて決定される。

エの外部設計はユーザから見た設計であり、内部設計はコンピュータから見た設計である。

例題演習

システムの外部設計を完了させるとき，承認を受けるものとして，適切なものはどれか。

ア　画面レイアウト　　イ　システム開発計画
ウ　物理データベース仕様　　エ　プログラム流れ図

解答解説

外部設計におけるデザインレビューに関する問題である。

アの画面レイアウトは外部設計、イのシステム開発計画は基本計画、ウの物理データベース仕様は内部設計、エのプログラム流れ図はプログラム設計またはプログラミングとなる。

外部設計終了段階で承認を受けるのは画面レイアウトで、求める答えはアとなる。

例題演習

外部設計の成果物に基づいて，実現方法や処理効率を考慮しながら，システム開発者の立場から進める設計作業はどれか。

ア　画面フロー設計　　イ　機能分割・構造化
ウ　コード設計　　エ　論理データ設計

解答解説

内部設計に関する問題である。

外部設計の成果物に基づいて、システム開発者の立場から進める設計作業であるから、内部設計作業である。アの画面フロー設計、ウのコード設計、エの論理データ設計は外部設計の作業であり、イの機能分割・構造化が内部設計の作業である。求める答えはイとなる。

例題演習

外部設計工程の論理データ設計で行うべき作業項目はどれか。

ア　データ項目の洗い出しとデータ構造の決定
イ　データファイル編成と媒体の決定
ウ　データへのアクセス時間とデータファイルの容量の見積り
エ　データレコードのレイアウトの決定

解答解説

論理データ設計に関する問題である。

論理データ設計で行う作業は、データ項目の洗い出し、データの関連性の分析、ファイル候補の決定、データ項目と処理の対応確認、ファイル仕様の作成である。

アは論理データ設計、イ、ウ、エは物理データ設計である。求める答えはアとなる。

例題演習

システム開発の内部設計工程で行う作業項目として、適切なものはどれか。

ア　コード設計　　イ　物理データ設計
ウ　プログラム構造化設計　　エ　論理データ設計

解答解説

内部設計工程の作業項目に関する問題である。

内部設計で行う作業

① 機能分割・構造化は、プログラム詳細機能の洗い出し、図式目次・総括ダイヤグラム・詳細ダイヤグラムの作成、プログラム間インタフェースの作成、プロセスフローの決定などを行う。

② 物理データの設計は、主要ファイルの項目／編成方式の決定、レコードレイアウトの設計などを行う。

③ 入出力詳細設計は、原票設計、帳票設計、画面設計、入力チェック方式、データ修正法、メッセージの設計などを行う。

④ 内部設計書の作成

アのコード設計、エの論理データ設計は外部設計、ウのプログラム構造化設計はプログラム設計、イの物理データ設計は内部設計で行う。求める答えはイとなる。

例題演習

内部設計段階の物理データ設計において、実施する項目はどれか。

ア　アクセス時間と容量の見積もり　　イ　データ項目の洗い出し
ウ　データの関連性の分析　　エ　ファイル仕様の作成

解答解説

内部設計の物理データ設計に関する問題である。

物理データ設計の作業内容

① ファイル特性の理解　　② ファイル編成方式の決定
③ ファイル媒体の決定　　④ レコードレイアウトの決定
⑤ アクセス時間と容量の見積

イのデータ項目の洗い出し、ウのデータ関連性の分析、エのファイル仕様の作成は外部設計の論理データ設計で行う。

アのアクセス時間と容量の見積は内部設計の物理データ設計で行う。求める答えはアとなる。

例題演習

プログラム間のインタフェースの決定において、明確にするのは、次のうちのどれか。

ア　業務フローを明確にする。　　イ　システムフローを明確にする。
ウ　プロセスフローを明確にする。　　エ　データフローを明確にする。

解答解説

プログラム間のインタフェースの決定に関する問題である。

アの業務フローは、業務処理の流れを示すもので、インタフェースには伝票・帳票を用いる。

イのシステムフローは、業務の流れをコンピュータシステム化した場合の処理の流れを表すもので、インタフェースには帳票類に相当する情報が利用される。

ウのプロセスフローは、業務処理の流れをプログラムの処理順序で表したもので、ファイルやデータの受け渡しを明らかにする。インタフェースには媒体が利用される。

エのデータフローは、データ処理の流れを表現したもので、源泉・吸収や処理プロセス、データの保存、取り扱うデータの関係が表現され、インタフェースはデータになる。

プログラム間のインタフェースの決定は、プログラム間でのデータの受渡を明確にすることであり、データフローを明確にすることである。求める答えはエとなる。

例題演習

ある販売店では，年間の購入実績によって客層を区分し，この客層区分に従って割引率を設定している。１年間の販売実績が売上日の順に次のような形式のレコードで記録されている。そのファイルに基づいて会計年度末に客層区分の見直しを行っている。その際に必要となる帳票の作成方法として，適切なものはどれか。

売上日	顧客 ID	客層区分	割引率	商品 ID	希望販売価格	販売数量	希望販売価格合計	販売金額

注　希望販売価格合計＝希望販売価格×販売数量
　　販売金額＝希望販売価格合計×（1－割引率）

ア　売上日をグループキーとして販売金額の集計を行い，販売金額を降順に帳票に印字する。
イ　客層区分をグループキーとして希望販売価格合計の集計を行い，希望販売価格合計の集計値を降順に帳票に印字する。
ウ　顧客ＩＤをグループキーとして希望販売価格合計の集計を行い，希望販売価格合計の集計値を降順に帳票に印字する。
エ　販売金額をグループキーとして販売金額の集計を行い，販売金額の集計値を降順に帳票に印字する。

解答解説

客層区分の見直しに必要な帳票の作成要領に関する問題である。

アの売上日をグループキーとして販売金額を集計しても、顧客ＩＤ別の情報を得ることができない。

イの客層区分別に希望販売金額を集計しても、顧客別の評価を適切に行うことができないため適正な客層区分の修正にはならない。

ウの顧客ＩＤ別に希望販売価格のグループ集計を行い、希望販売価格の合計の集計値を求め、降順に整列すると適正な評価が可能になる。求める答えはウとなる。

エの販売金額別に販売金額を集計しても、顧客別の情報は不明である。

例題演習

Ａ社では，優良顧客の層について調査することになった。優良顧客とは，最近購入実績があり，かつ購入回数の多い人とする。優良顧客の絞り込みを行うため，最近の１か月，２か月，３か月，… について，期間ごとに購入回数ごとの顧客数を数え，分析表を作成することにした。優良顧客の層を求めやすい適切な分析表はどれか。

ア

顧客購入分析表

月＼回数	10	9	…
最近１か月	550	650	…
最近２か月	700	850	…
⋮	⋮	⋮	⋮

イ

顧客購入分析表（月）

最近１か月	最近２か月	最近３か月	…
3,500	3,800	4,200	…

顧客購入分析表（回数）

10	9	8	…
2,000	2,500	2,800	…

ウ

顧客購入分析表

回数	月	顧客数
10	最近１か月	550
	最近２か月	700
	⋮	⋮
9	最近１か月	650
⋮	⋮	⋮

エ

顧客購入分析表

月	回数	顧客数
最近１か月	10	550
	9	650
	⋮	⋮
最近２か月	10	700
⋮	⋮	⋮

解答解説

解析条件を満足させる分析表の作成に関する問題である。

優良顧客の定義は、最近購入実績があり、購入回数の多い人である。

判断データは最近の１ヶ月、２ヶ月、…別に期間ごとの購入回数を求めて判定する。期間ごとの購入回数が一目で分かるようになっている表が全体情報と部分情報を同時に見ることができ、適切な判断につながる情報になる。

アの表は、マトリックス上に最近の期間と購入回数が整理されており、期間と購入回数の対応付けが容易であり適切な判断ができる表現である。求める答えはアとなる。

イの情報は、最近の期間と購入回数の関係づけが不十分である。

ウの情報は、購入回数から最近の期間を見る場合には利用できるが、購入回数に最近の期間を加味して考える場合に使用し難い表である。

エの場合はウの場合の逆の現象である。購入回数と最近の期間を関係づけて判断する場合に使用し難い情報提供法である。

例題演習

ある企業では，顧客マスタファイル，商品マスタファイル，担当者マスタファイル及び当月受注ファイルを基に月次で受注実績の把握を行っている。各ファイルの項目が表のとおりであるとき、これら四つのファイルを使用して当月と前３か月分の出力が可能な受注実績はどれか。

ファイル	項　　目	備　考
顧客マスタ ファイル	顧客コード，名称， 前月受注額，2か月前受注額，3か月前受注額	―――――
商品マスタ ファイル	商品コード，名称， 前月受注額，2か月前受注額，3か月前受注額	―――――
担当者マスタ ファイル	担当者コード，氏名， 担当顧客数分の顧客コード	1顧客の担当者は1人
当月受注 ファイル	顧客コード， 商品コード，受注額	―――――

ア　顧客別の商品別受注実績　　イ　商品別の顧客別受注実績

ウ　商品別の担当者別受注実績　　エ　担当者別の顧客別受注実績

解答解説

マスタファイルの利用に関する問題である。

顧客マスタファイル、商品マスタファイル、担当者マスタファイル、当月受注ファイルを使用して、当月と前３ヶ月の受注実績を求める問題である。

ア、イは、前３ヶ月の顧客と商品を関連づけるデータがないため不能である。

ウは、商品と担当者を関連づけるデータがないため不能である。

エは顧客コードから担当者コードを決定することが可能であり、当月と前３ヶ月の担当者別の顧客別受注実績を顧客マスタ、担当者マスタ、当月受注ファイルから作成可能である。求める答えはエである。

例題演習

様式が複数ある伝票から，様式ごとに定められた項目のデータを入力する。入力漏れがないことを確認するためにプログラムで最初に実行すべき処理として，適切なものはどれか。

ア　伝票の様式ごとに定められた項目数と入力された項目数を比較する。

イ　伝票の様式を示すコードを使って，入力された項目のデータ形式を検査する。

ウ　入力された項目の内容が，伝票の様式で指定されたデータ形式と一致していることを確認する。

エ　マスタファイルと突き合わせて，入力された項目の内容が正しいことを確認する。

解答解説

入力漏れのチェックに関する問題である。

アの定められた項目数と入力された項目数の比較は入力漏れのチェックになる。求める答えはアとなる。

イの入力された項目のデータ形式の検査は入力内容のチェックであって、入力漏れのチェックではない。

ウのデータ形式の確認は入力漏れのチェックにはならない。

エのマスタファイルの突き合わせは、内容のチェックであって入力漏れのチェックではない。

例題演習

状態遷移図を用いて設計を行うことが最も適しているシステムはどれか。

ア　月末及び決算時の棚卸資産を集計処理する在庫棚卸システム

イ　システム資源の稼働状態を計測し，レポートとして出力するシステム資源稼働状態計測システム

ウ　水道の検針データから料金を計算する水道料金計算システム

エ　設置したセンサの情報から，温室内の環境を最適に保つ温室制御システム

解答解説

状態遷移図に関する問題である。

状態遷移図は、時間の経過や状況の変化に基づいて、その時々の動作や状態を記述するものであるから、室内の温度を計測するセンサからの情報に基づいて、室内の環境を最適に保つ温室制御を表すために用いられる。求める答えはエとなる。

例題演習

ソフトウェアの分析・設計技法のうち，データ中心分析・設計技法に関する特徴を記述したものはどれか。

ア　システム開発後の仕様変更は，データ構造や手続きを局所的に変更したり，追加したりすることによって比較的容易に実現できる。

イ　対象業務領域のモデル化に際して，最も安定した情報資源に着目する。

ウ　プログラムが最も効率よくアクセスできるようにデータ構造を設計する。

エ　モジュールの独立性が高くなるようにプログラムを分割し，機能を詳細化していく。

解答解説

データ中心分析・設計技法の特徴に関する問題である。

アは運用保守を効率的に推進するソフトウェアの保守に対する考え方である。

イの対象業務のモデル化とその情報資源の活用がデータ中心分析・設計手法の基本的な考え方で、求める答えはイとなる。

ウはデータへのアクセスを効率的に行うデータベース設計の考え方である。

エはモジュールの独立性を高めソフトウェアの再利用を高める考え方である。

例題演習

データ中心アプローチに基づくデータ標準化作業の手順として、適切なものはどれか。

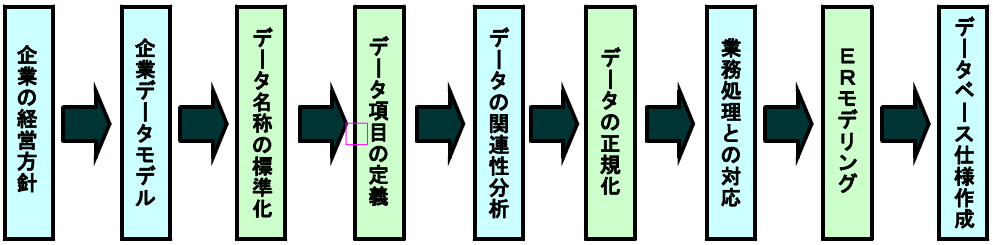
ａ．Ｅ−Ｒモデリング　　ｂ．データ項目定義
ｃ．データ正規化　　ｄ．データ名称の標準化

ア　ｂ−ｃ−ｄ−ａ　　イ　ｂ−ｄ−ｃ−ａ　　ウ　ｄ−ｂ−ｃ−ａ　　エ　ｄ−ｃ−ｂ−ａ

解答解説

データ中心アプローチに基づくデータ標準化作業の問題である。

データ標準化、データベース設計の手順



この問題は正規化の手順を求める問題であり、

データ名称の標準化→データ項目の定義→データの正規化→Ｅ−Ｒモデリング

の順になり、求める答えはウとなる。

例題演習

データ中心アプローチによる開発手順について、次の作業項目を正しい順序に並べたものはどれか。

（１）　応用プログラム設計　　（２）　カプセル化
（３）　データモデリング　　（４）　ドメイン／原子オブジェクト分析

ア　（３）（４）（１）（２）　　イ　（３）（４）（２）（１）
ウ　（４）（３）（１）（２）　　エ　（４）（３）（２）（１）

解答解説

データ中心アプローチの開発手順に関する問題である。

最初にデータベースの構築であり、データベース構築ではデータの正規化が重要テーマである。企業モデルを分析し、データ名称の標準化、データ項目の定義、関連性の分析、データの正規化が行われる。更に、業務処理との対応を検討し、ＥＲモデリングを作成する。

データモデリング→ドメイン／原始オブジェクト分析→カプセル化→応用プログラム設計の

手順になる。求める答えはイとなる。

例題演習

データ中心アプローチに関する記述のうち、最も適切なものはどれか。

ア　データ資源の重複だけでなく、それに起因するプロセスの重複も排除することを目的としている。
イ　データとその処理手順のカプセル化に見られるように、オブジェクト指向の方法論をデータベース設計に応用しょうとする試みである。
ウ　データの流れに着目してシステム分析を行い、再利用可能なモジュールを抽出することによってソフトウェアの生産性を向上させることを目標としている。
エ　データベースの最適設計のため、プログラム設計については、ＤＦＤによって抽出されたプロセスに対して構造化技法を併用する必要がある。

解答解説

データ中心アプローチに関する問題である。

データ中心アプローチは、データや情報を組織体の共有の経営資源と見なし、他の経営資源と同様に計画・管理・統制していく考え方である。データや情報資源をシステムの骨組みと考え、対象業務が持つデータとそれらの関連やデータの発生・流通の仕組みを定義していく。

アはデータ中心アプローチ、イはオブジェクト指向、ウはモジュール分割の考え方、エは機能中心アプローチである。求める答えはアとなる。

例題演習

プロセス中心設計と比較したとき，データ中心設計の特徴として，適切なものはどれか。

ア　業務のモデリングに先だって，データモデリングを行う。
イ　業務プロセスに合わせたデータ構造が作成できる。
ウ　データを共有資源と見なし，一元的に管理できる。
エ　特定の業務に関する短期間のシステム化に有効である。

解答解説

データ中心設計の特徴に関する問題である。

データ中心アプローチは、データや情報を組織体の共有の経営資源と見なし、他の人、もの、金と同様に計画・管理・統制していこうという考え方である。データをシステムの骨組みと考え、対象業務がもつデータとそれらの関連やデータの発生・流通の仕組みを定義していく。

データ中心アプローチの設計手順は次の通りである。

① 企業目標の設定　② 企業活動の分析
③ 情報システムの体系化　④ データ体系の確立
⑤ データモデルの作成　⑥ 個別業務システムの開発

アはデータ中心設計の設計手順、イはプロセス中心設計の特徴、ウはデータ中心設計の特徴、

エはプロセス中心アプローチの考え方の特徴である。求める答えはウである。

例題演習

データ中心アプローチ（ＤＯＡ）の特徴として、適切なものはどれか。

ア　広範囲のデータを体系化することは多大な労力がかかるので、開発効率を向上させるために、特定業務に関連するデータを体系化するアプローチである。

イ　システム全体の整合性を保ち、設計の修正を最小限とするために、機能の設計から詳細設計へと進むアプローチである。

ウ　対象業務を独立性の高い単位に分割することによって、設計効率を向上させ、その分割された単位で実装を行うアプローチである。

エ　データとデータ操作を一体化して標準部品とし、このような標準部品を利用してシステムを構成するアプローチである。

解答解説

データ中心アプローチに関する問題である。

データ中心アプローチの展開手順

①　企業目標の設定では、企業全体を対象にデータを分析する。

②　企業活動の分析では、企業目標を達成するために必要な活動を明確にし、ビジネスプロセスとデータクラスの関連を明確にする。

③　情報システムの体系化では、ビジネスプロセスとデータクラスの関連を分析して、情報システム体系を明確にする。

④　データ体系の確立では、データクラスをブレークダウンして、管理対象であるエンティティを識別し、企業データモデルを作成する。

⑤　データモデルの作成で、企業データモデルに基づいて個別業務のデータモデルを作成する。

⑥　個別業務システムの開発では、データモデルを前提にして個別業務システムを開発する。

アは、特定業務に関連するデータの体系化が問題である。

イは、機能設計から詳細設計へ進むアプローチが問題である。

ウは、独立性の高い対象業務の分割から実行しているのが問題である。

エのデータとデータ操作を一体化した標準部品を利用してシステムを構成する考え方がデータ中心アプローチである。求める答えはエとなる。

txt040212 コードの設計

①　コード化の目的

ⓐ　コード化とは

コンピュータで効率よく処理するためには、情報のコード化を適切に行う必要がある。コードとは本来の名称を使用目的に応じて、識別するための略号や記号、符号、暗号などのことである。入出力データとして使用するコードは外部コードであり、外部設計工程で行うコード設計は外部コードである。

ⓑ　コード化の必要性

㋐　記号を定めて統一的に管理するためコード化を行う。
㋑　コードで入力すると、入力速度の向上を図ることができる。
㋒　分類を識別し、集計することが可能になる。
㋓　コード化はコンピュータ処理上の有効な手段である。

ⓒ　良いコードの条件

㋐　扱いやすいこと

人間が扱いやすいコードにするためには、コードの割り当て、コードの転記、コードのチェックの検討が必要になる。短い固定の桁数、単純なものがよい。４桁程度がよいが、長くなる場合にはハイフンで区切って使用する。数字と英字だけで構成されることか望ましい。漢字はコードには適さない。設定の考え方が標準化されている必要がある。コンピュータの処理に適したコードの条件も重要である。

㋑　共通性のあること

コードはすべての業務で共通に使用できることが望ましい。社内は同一コードを使用すると共に、取引先のコード体系も同一にする。異なるコード体系を使用するときは、システムの入り口で変換する。

㋒　体系的であること

体系化はグループ分けを可能にすることである。グループ分けができるのは、コードに規則性がある場合である。上位の桁から、大分類、中分類、小分類と意味を与える。体系化すると、各レベルの合計の計算が容易になり、コードの追加も容易になる。

㋓　拡張性があること

コード化対象の増減があることを前提に考える。新規コードの割り当ては、現行コードの

末尾に追加する方法と適当な位置に追加する方法がある。単純な連番コードは末尾に追加できるが、グループ化したコードではコード番号に余裕を持たせる必要がある。

㋔　明瞭性があること

コードは数字を原則とし、必要に応じて英字や仮名を組み合わせる。誤りやすい文字の組み合わせは避ける。記入ミスや誤読が発生しやすいものがあるので、採用するときには慎重に検討する。特に、数字と英字を混合してコードにする場合には、注意する必要がある。

ⓓ　コード機能の分類

㋐　識別機能

他のデータと区別するための機能である。同姓同名の人がいる場合、コード化すると識別することができる。

㋑　分類機能

対象をグループ化するための機能である。

㋒　配列機能

データの並びの順を決める。

㋓　チェック機能

コンピュータにコードを使用して入力する際に、入力されたデータが正しいかどうかのチェックを可能にする機能である。コードの最下位にチェックディジットコードを付加し、決められた計算式に従って計算し、チェックディジットと一致すると、正しく入力されたと判断する。

ⓔ　主要なコードの種類と特徴

㋐　連番法

コードの対象を順番に並べて先頭から番号を付ける方法である。代表的な連番コードに、ＪＩＳの都道府県コードがある。北海道を01として、沖縄県を47とする連番が北から南に向かって付けられている。

㋑　区分分類法

コード化対象データをある共通の特性に基づいて、任意の大きさのブロックに区分けして、各ブロック内に順番号を割り振る方式である。区の番号、市の番号、町村の番号などの分類方法に用いられている。

㋒　桁別分類法

一定の基準で、大分類・中分類・小分類と階層的に分類し、おのおののグループ単位に連番を割り振る方式である。企業内の組織コードに使用される。支店コード、部コード、課コードと階層的に分類する。

㋓　10進分類法

10進数の原則に従って、全体を10分割し、更にそれを10分割するように、10分割を繰り返す方法である。図書館の図書分類に用いられている。

㋔　表意法

データ自体の名称などをコードに何らかの形で組み込み、コードから対象データを容易に連想できるようにしたものである。製造年月日や入学年度の数字をそのままコードの中に入れたりする。

㋕　合成法

各種の方法を適宜組み合わせたものである。

種類	長　所	短　所
連番法	単純である。 一意性が確立されている。	内容の連想が困難である。 分類ができない。 途中の追加ができない。
区分分類法	データ件数の偏りに対処できる。	不規則な分類である。 機械処理が複雑になる。
桁別分類法	分類基準が明確である。 追加補充に融通性がある。 コードの意味が理解しやすい	桁数が多くなる。
10進分類法	無限に追加できる。	桁数が多くなる。 桁数が不揃いになる。
表意法	コードが記憶しやすい。	桁数が多くなる。 シノニムの発生が高い。
合成法	各種コードの利点が利用できる。	コード体系が複雑になる。 桁数が多くなる。

②　コード設計

ⓐ　コード設計の内容

㋐　業務コード、取引先コード等一目で分かる名称を付ける。

㋑　商品コード、金融機関コード等の一般に使用されている共通コードの利用を検討する。
㋒　コードの各桁の意味や将来の拡張性等などを検討する。

ⓑ　コード設計上の留意点

㋐　扱いやすさと標準化
㋑　共通性
㋒　体系化
㋓　拡張性
㋔　明瞭性

ⓒ　コード設計の手順

㋐　コード化対象の選定

コード化は将来にわたって、対象項目の管理を効率的に行うために必要で、標準化や体系化の観点からも検討が必要である。現状調査と分析を通じて、コード化すべき項目の候補をあげ、システム処理上の効率なども加味して、コード化すべき項目を決定する。すでにコード化されているデータ項目についても、機械化処理する上で問題がないかを検討する。

㋑　コード化の目的の明確化

識別、分類、配列、チェックいずれの目的に使用するのかを明確にする。コード化の目的は、コード設計の重要な判断基準になる。

㋒　コード化データ量の予測

コード化対象項目について、現在の項目数の調査とコードの使用期間において増減する数を予測する。現在、必要とする件数ではなく、コードの使用期間、データ件数の成長率などを考慮し、将来の件数を可能な限り精度高く予測する。コードの桁数の設定に関係する。

㋓　利用範囲決定と使用期間推定

利用範囲は広範囲にわたるのが望ましいが、広範囲になりすぎると調査が膨大になり、決定が困難になる。利用期間も長期にわたるのが望ましいが、長期になりすぎると対象の増減が難しくなる。範囲と期間を限定しすぎると、コード改正の機会が多くなる。コード改正の影響は大きく、それに要する労力も費用も膨大になる。十分に吟味し、拡張性をもったコード体系を確立する。

㋔　コード化対象データの特性分析

コードの使用範囲、付番対象データのばらつき具合、コード変更の周期と頻度、外部組織との関連、マンマシンインタフェースなどを考慮し、最適の付番方式を決定する。コード体系に柔軟性がないと、開発されたシステムのライフサイクルが短くなる。

㋕　コードの考案

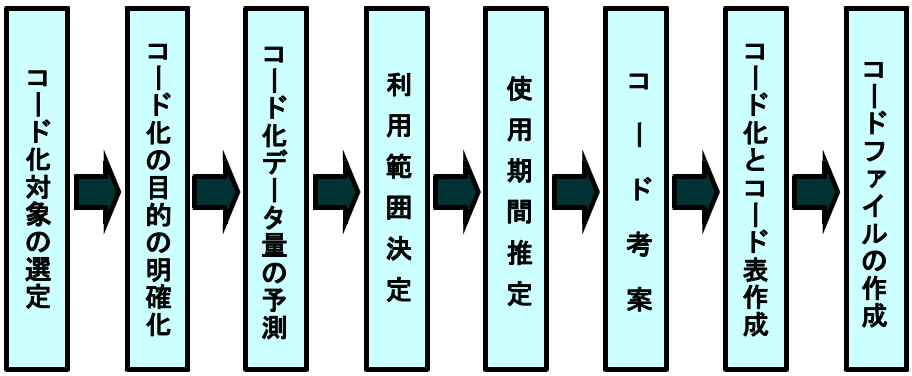
コードの種類と特徴を調査し、具体的なコード体系を考案する。コード体系、コードの桁数、コードを付与するための基準、チェックが必要なコード、設計したコードが本来の目的に合致しているかの検討などを行う。

㋖　コード化作業とコード表の作成

コード設計書でコード体系やコード名、意味、用途、桁数、属性などを定義したら、コード表を作成する。コードの値を設定するとき、追加の可能性を考えて、拡張を考慮した決定をする。

㋗　コードファイルの作成

業務処理に使用するため、コード表の内容を必要な媒体に記録する。各種のコード表を集めて１冊にしたものをコードブックといい、コードの保守用に用いる。また、コード表と合わせて実際の業務に使用する。コードは追加や変更が発生しやすい。拡張性や変更容易性が重要で、そのためにもコードブックの整備が不可欠である。コードブックには、コード体系、コード名、意味、用途、桁数、属性、コード値などの定義した内容を記載する。



③　コードのエラーチェック

ⓐ　コード入力エラーの種類

㋐　トランスクリプションエラー

文字の書き違い。

具体例　　12345　→　12845

㋑　トランスポジションエラー

文字の入れ違いによる間違いや１桁おいた文字の入れ違いはダブルトランスポジションエラーがある。

具体例

トランスポジションエラー　　12345　→　13245
ダブルトランスポジションエラー　　12345　→　14325

㋒　ランダムエラー

㋐と㋑のエラーが重なった場合やそれ以外のエラーである。

具体例　　12345　→　14325

ⓑ　基本的なエラーの防止方法

㋐　数字の桁数が長くなるのを防ぐ

長くなるときは、３桁または４桁毎にハイフンを入れる。４桁を越えると、人間の記憶の正確さが急減する。

㋑　英字と数字を組み合わせる

単調なコードにならないようにする。

ⓒ　チェックディジット法

㋐　基本ステップ

❶　コードの各桁別にウェイトを決める。各種ウェイトが用いられる。
❷　各桁と各ウェイトを乗算して和を求める。積和法と桁分離積和法がある。
❸　モジュラスにより余りを求める。モジュラス10とモジュラス11がある。
❹　チェックディジットを決める。余りを使う方法とモジュラスから余りを引く方法がある。

㋑　ウェイト

ウェイトの用い方で検出力が異なる。

ウェイトの例

（１、２、１、２、１、２）、（１、３、１、３、１、３）
（１、２、３、４、５、６）、（７、６、５、４、３、２）

㋒　積和法

コードとウェイトを桁別に乗算して、その和を求める。乗算した結果の数値が２桁になったら、その数値を分離して１桁の数値にして、合計を求める桁分離積和法がある。

㋓　モジュラス

モジュラスは除数の意味である。10を使う場合と11を使う場合がある。11を使う場合に余りが２桁になる場合がある。この場合、余りが10の場合は０にするとかＸを使うとかの約束が必要である。０の場合は検出力が低下する。

㋔　チェックディジット

❶　余りをチェックディジットとする。

❷　モジュラスから余りを引いたものをチェックディジットとする。

ⓓ　チェックディジット法の特徴

㋐　コードの末尾に人為的に数字を付加して、入力誤りの検出に利用する。

㋑　対象とするエラーの種類としては、文字の誤り、隣接文字の逆転誤り、１桁挟んだ文字の逆転誤りなどがある。

㋒　使用するウェイトによって、エラーの検出力が異なる。

㋓　ランダムエラーの検出力は低い。

ⓔ　チェックディジット法の具体例

コード	１	２	３	４	５
ウェイト	５	９	１	７	３
乗算と合計	５＋	18＋	３＋	28＋	15＝69

モジュラス１１を使用　　69÷11＝６　余り　３

この余りをチェックディジットコードとして、末尾に付ける。

　　1234５3　　下線の部分がチェックディジット

余りからさらに除数の補数を求める方法もある。11－３＝８

　　1234５8　　下線の部分がチェックディジット

ⓕ　チェックディジット法の利点

㋐　エラーがコンピュータ内部の計算によって発見できる。

㋑　各桁のウェイトと計算式を機密化することで不正利用を防止できる。

㋒　ウェイトを変えることで防犯体制を変更できる。

ⓖ　チェックディジット法の検出力

㋐　入力エラーの文字の書き違いや文字の入れ違い(トランスポジションエラー)はかなりの確率で検出できる。

㋑　ランダムエラーや入力エラーの文字の書き違いや文字の入れ違い(トランスポジションエラー)が重なったエラー等は検出に限界がある。

㋒　他のチェック方法との併用が必要である。

④　データのエラーチェック

ⓐ　入力段階に発生するエラー

㋐　データ自体の誤り

原票の記載内容の誤りや入力時の操作ミスによる誤りで、一番発生確率の高い誤りである。

㋑　データの過不足

データの収集段階に発生する誤りで、原票の紛失や脱落あるいは重複などによって、データ件数が誤りになる。

㋒　データの遅延

データの収集段階で発生する誤りで、時間的に間に合わなかったデータがあることによって発生する誤りである。

ⓑ　エラーの修正方法

㋐　原票のエラーの修正方法

原票を起票した部門に差し戻して、正しく記入してもらう。業務の内容を知らずに、経験と勘で修正すると誤った修正をする。

㋑　プログラムによるエラーの検出時の修正法

エラーがなくなるまで入力データをチェックし、すべてのデータが正しくなった時点で処理を行う。データの修正に時間がかかりすぎる問題があり、不定期業務やデータ量の少ない業務に適している。

エラーデータを除外して処理を行う。エラーとなったデータは、修正後、次のデータ入力時に一緒に入れる。比較的処理サイクルの短い業務で、データの修正を待っていると、時間的余裕がない場合に適している。正しいデータは一部分処理を進めておき、エラーデータは修正後、正しいデータと併合する。

全データを一本化してから本来の処理を行う。正しいデータだけで処理を行い、エラーデータは修正後別データとして処理する。処理が途中で停止しないので、操作が簡単になる。エラーデータは、元々なかったものとして処理する。大量データの統計処理などで多少のエラーデータがあっても結果に影響がない場合に用いる。

⑤　プログラムによるチェック

ⓐ　エラーチェックのタイミング

データのエラーは入力段階でチェックする。入力されたデータを、入力時点でチェックし、エラーデータをシステムに潜り込ませないようにすることが重要である。

ⓑ　プログラムによる入力データのチェック法

㋐　数字チェック

計算に使用する項目に数字が入っているかをチェックする。数字以外の文字が入っていると、計算するときにデータエラーとなる。

㋑　形式チェック

データの各項目の桁位置、桁数が決められた形式で入力されているかをチェックする。

㋒　限界チェック（範囲チェック）

データの値が、下限値と上限値の間にあるかを検査する。

㋓　論理チェツク（妥当性チェック）

項目の内容が、決められている値のいずれかに該当するかを検査する。昭和は元年から６３年までなので、それ以外のデータはエラーとなる。

㋔　照合チェック

入力された得意先コードや商品コードが実在するかどうかを得意先マスタファイルや商品マスタファイルと照合してチェックする。

㋕　シーケンスチェック

データの順序が決められた通りかどうかを検査する。

㋖　バランスチェック

借方・貸方のように、対になる項目の数値や合計が等しいかどうかを検査する。

㋗　クロストータルチェック

横計、縦計が一致するかどうかを検査する。

㋘　バッチトータルチェック

あらかじめ手作業で計算しておいた合計値を、データと一緒に入力し、コンピュータ内で計算したデータの合計値と一致するかを検査する。

㋙　ハッシュトータルチェック（ごた混ぜ合計）

各レコード毎にレコード番号、数量、単価などを手作業で合計して、データと一緒に入力し、コンピュータ内部の計算結果とチェックする。

㋚　カウントチェック

レコード件数、原票枚数などを、入力の前後でカウントして照合する。

㋛　重複チェック

許されない重複データがあるかどうかをチェックする。

㋜　リダンダンシーチェック

パリティビットを用いてチェックを行う。

㋝　チェックディジットチェック

チェック用の数字を付加してチェックを行う。

例題演習

モジュラス11などの計算方法によって得られた結果を商品コードなどの末尾に付加し，入力の誤りを入力データだけから発見できるようにする方法がある。この末尾に付加されるものを何というか。

ア　けた別コード　　イ　チエックディジット
ウ　チエックポイント　　エ　デシマルコード

解答解説

チェックディジットに関する問題である。

アの桁別コード、エのデシマルコードはコード分類の一種類である。

イの入力データの誤りをチェックするために一定の計算法によって数値を算出して利用するのは、チェックディジットの方式である。求める答えはイである。

ウのチェックポイントは、プログラム実行中のある時点で、リスタートのための情報を生成する点である。チェックポイントを通過する時点の主記憶の情報を磁気ディスクに出力しておき、万一システムが停止しても、チェックポイントから処理を継続できるようにする。

エのデシマルコードは、10進分類法である。

例題演習

業務システムのコード設計に関する記述のうち，最も適切なものはどれか。

ア　コードの実際の付番は，コードの処理方法に詳しいシステム設計担当者が行うべきである。
イ　コードの属性とけた数は，コンピュータの内部処理効率に重点を置いて設計すべきである。
ウ　コードの入力ミスが業務に重大な影響を及ぼすと判断されるときは，検査文字（チェックディジットなど）を採用すべきである。
エ　コードの保守方法（追加，廃止，変更など）については，運用テストの段階で決めるべきである。

解答解説

コード設計のチェックディジット法に関する問題である。

チェックディジットは、コード入力誤りをチェックするために、コードに付加する検査文字で、入力されたコードから計算されたチェックディジットとコードに付加されたチェックディジットを照合してエラーを検出する。

アのコードの実際の付番はコードの処理方法に詳しいシステム利用者が行う方がよい。システム設計者は必ずしも詳しいとは言えない。

イのコードの属性と桁数はコードの使用条件に従って設計すべきである。コンピュータの内部処理効率ではない。

ウのコードの入力ミスを防止するためには、検査文字を採用すべきである内容は適切な記述である。求める答えはウとなる。

エのコードの保守方法の検討は外部設計段階で決めるべきである。運用テスト段階では遅い。

例題演習

次の方式によって求められるチェックディジットを付加した結果はどれか。

ここで，データを7394，重み付け定数を1234，基数を11とする。

〔方式〕

（１）データと重み付け定数の各けたの積を求め，その和を求める。

（２）和を基数で割って，余りを求める。、

（３）基数から余りを減じ，その結果の１の位をチェックディジットとしてデータの末尾に付加する。

ア　73940　　イ　73941　　ウ　73944　　エ　73947

解答解説

チェックディジットに関する問題である。

チェックディジットの計算要領

①　コードの各桁に指定したウェイトを掛ける

②　①で求めた結果の和を求める。(桁別分離法とそのままで和を求める場合がある)

③　②で求めた結果をモジュラス(10または11)で割り、余りを求める。

④　③で求めた余りまたはモジュラスから余りを引いて、検査文字を求める。

チェックディジットコードは、次のように計算する。

７×１＋３×２＋９×３＋４×４＝７＋６＋27＋16＝56

56÷11＝５余り１

11－１＝10

従って、余り10の１の位であるから、検査文字は０である。求めるコードは73940となり、求める答えはアとなる。

例題演習

顧客コードにチェックディジット(検査数字)を付加する目的として，適切なものはどれか。

ア　顧客コードの入力誤りを発見する。

イ　顧客名簿を作るときに，獲得した順に顧客を配列する。

ウ　顧客を地区別などのグループに分類できるようにする。

エ　特定の顧客を類推できるようにする。

解答解説

チェックディジットコードに関する問題である。

チェックディジットによるチェック要領

①　数字項目の各けたに重み付けを行い、ある計算して得られた１けたの数字をチェックディジットとして数字項目に付加する。

②　同じ計算を行って得られた数字とチェックディジットの値を比較し、一致すれば正しいデータ、一致しなければ間違ったデータと判断する。

アの顧客コードの入力誤りのチェックに使用するのはチェックディジッドコードを付加する目的である。求める答えはアとなる。

イの配列に使用する、ウの分類に使用する、類推に使用するはいずれも誤りである。

例題演習

４桁のコードＮ１Ｎ２Ｎ３Ｃがある。最右端のＣはチェックディジットであり、次の方法で計算する。

Ｃ＝ｍｏｄ（(Ｎ１×３＋Ｎ２×２＋Ｎ３×１)、10)

ただし、ｍｏｄ（ａ、ｂ）はａ／ｂの剰余とする。

次の４桁のコードの□にあてはまる数字はどれか。

８１□６

ア　０　　イ　２　　ウ　４　　エ　６

解答解説

チェックディジット付きコードを求める問題である。

アの場合、810 を計算すると、24＋２＝26　mod(26,10)＝６　求める答えはアである。

イの場合、24＋２＋２＝28　mod(28,10)＝８　当てはまらない。

ウの場合、24＋２＋４＝30　mod(30,10)＝０　当てはまらない。

エの場合、24＋２＋６＝32　mod(32,10)＝２　当てはまらない。

例題演習

与えられたデータから一定の規則に従って数値を算出し，この数値から検査文字を定めて与えられたデータに付加する。これを用いて入力データの検査を行う。次の規則を用いた場合に，４けたの数値データ“2131”に付加する検査文字として，正しいものはどれか。

〔規則〕

（１）与えられたデータの各けたに，先頭から係数４，３，２，１を割り当てる。

（２）各けたの数値と割り当てた係数とのそれぞれの積の和を求める。

（３）（２）で求めた和を11で割って余りを求める。

（４）（３）で求めた余りの数字を検査文字とする。ただし，余りが10のときは，“Ｘ”を検査文字とする。

ア　１　　イ　３　　ウ　５　　エ　７

解答解説

チェックディジット法に関する問題である。

数値データ2131に付加する検査文字を求める。

２×４＋１×３＋３×２＋１＝18

18／11＝１……７

検査文字は７となる。求める答えはエとなる。

例題演習

０～６の数４個で構成される数列(N₃, N₂, N₁, C)がある。Cはチェックディジット(検査数字)であり,

C＝(N₃×３＋N₂×２＋N₁×１) mod 7

を満たす。数列(４, ２, □, ６)がこの条件を満たすとき, □に当てはまる数はどれか。

ここで, a mod bは, aをbで割った余りを表す.

ア ０　　イ ２　　ウ ４　　エ ６

解答解説

チェックディジットに関する問題である。

□の数字をXとすると、次の式が成り立つ。

(４×３＋２×２＋X) mod 7＝６

(12＋４＋X) mod 7＝６

アの場合、16 mod 7＝２

イの場合、18 mod 7＝４

ウの場合、20 mod 7＝６　求める答えはウとなる。

エの場合、22 mod 7＝１

例題演習

バーコードには, 検査数字(チェックディジット)を付加するのが一般的である。JANコード(標準タイプ, 13けた)では, 12けたの数の検査数字を次の方式で算出している。この方式で算出した図のバーコード(123456789012)の検査数字として適切な値はどれか。

〔JANコードにおける検査数字の算出及び付加方式〕

（１）検査数字を付加する前の右端の数字の位置を奇数けたとし, 左に向かって交互に奇数けたと偶数けたとする。

（２）偶数けたの数字の合計を求める。

（３）奇数けたの数字の合計を求め, その値を３倍する。

（４）(２)と(３)の合計を求める。

（５）(４)の値の１の位の数字を10から引く。ただし, １の位が０のときは０とする。例えば, (４)の値が123のときは10－３＝７, 120のときは０とする。

（６）(５)で求めた数字を検査数字とし, 右端けたの右に付加する。

ア ０　　イ ３　　ウ ５　　エ ８

解答解説

チェックディジットコードに関する問題である。

手順に従って、検査数字を求めると次のようになる。

偶数桁の合計　１＋３＋５＋７＋９＋１＝26

奇数桁の合計　２＋４＋６＋８＋０＋２＝22

奇数桁の合計を３倍する　　22×３＝66

全体の合計を求める　　26＋66＝92

検査数字を求める　10－２＝８

検査数字は８となり、求める答えはエとなる。

例題演習

チェックディジットを利用する目的として，適切なものはどれか。

ア　数値項目へ入力したデータに，英字や記号が混入した誤りを検出する。

イ　入力したコードの値の誤りを検出する。

ウ　入力したコードのけた数の誤りを検出する。

エ　入力したデータ値が，定められた範囲内に収まっていない誤りを検出する。

解答解説

チェックディジットに関する問題である。

チェックディジットは、データの項目から、ある規則に従って導かれる検査数字をそのデータ項目に付加することによって入力データの誤りをチェックする方法である。

アは数字チェック方式、イはチェックディジット方式、ウは形式チェック、エは範囲チェック方式である。求める答えはイとなる。

例題演習

入力データのチェック方式の一つであるリミットチェックの説明として、適切なものはどれか。

ア　重複したデータが存在するかどうかをチェックする。

イ　データが論理的に正しいかどうかをチェックする。

ウ　データの値が一定の範囲内にあるかどうかをチェックする。

エ　データのコードと正しいコードを記録した表とを照合し、チェックする。

解答解説

データの誤りチェックのリミットチェックに関する問題である。

リミットチェックは、限度検査、範囲検査のことで、データの値が指定された値を超えていないかどうかや、指定された範囲内にあるかどうかを検査する方法である。

アは重複チェック、イは論理チェック、ウはリミットチェック、エは照合チェックである。求める答えはウとなる。

例題演習

次の注文データが入力されたとき，商品コードが商品マスタファイル上に存在するかどうかを調べるときに行うチェックはどれか。

注文データ

伝票番号 （文字）	注文日 （文字）	商品コード （文字）	数量 （数値）	顧客コード （文字）

ア　シーケンスチエック　　イ　照合チェック

ウ　ニューメリックチエック　　エ　論理チェック

解答解説

商品コードの照合チェックに関する問題である。

アのシーケンスチェックは、データの順序が決められた通りかどうかを検査する。

イの照合チェックは、入力された得意先コードや商品コードが実在するかどうかを得意先マスターファイルや商品マスターファイルと照合してチェックする。求める答えはイとなる。

ウのニューメリックチェックは、計算に使用する項目に数字が入っているかをチェックする。数字以外の文字が入っていると計算するときにエラーとなる。

エの論理チェックは、項目の内容が決められている値のいずれかに該当するかを検査する。例えば、昭和の年数は１～63の範囲内にあるかどうかをチェックする。

例題演習

次のような注文データが入力されたとき，注文日が入力日以前の営業日かどうかを検査するために行うチェックはどれか。

注文データ

伝票番号 （文字）	注文日 （文字）	商品コード （文字）	数量 （数値）	顧客コード （文字）

ア　シーケンスチェック　　イ　重複チェック

ウ　フォーマットチェック　　エ　論理チェック

解答解説

注文日の論理チェックに関する問題である。

アのシーケンスチェックは、データの順序が決められた通りかどうかを検査する。

イの重複チェックは、許されない重複データがあるかどうかをチェックすることである。

ウのフォーマットチェックは、データの各項目の桁位置、けた数が決められた形式で入力されているかどうかをチェックすることである。

エの論理チェックは、項目の内容が決められている値のいずれかに該当するかを検査する。例えば、昭和の年数は１～６３の範囲内にあるかどうかをチェックする。注文日が入力日以前

の営業日かを検査するのは論理チェックである。求める答えはエとなる。

例題演習

入力データの値が規定の範囲内かどうかを検査するチェック方法はどれか。

ア　照合チェック　　イ　重複チェック
ウ　フォーマットチェック　　エ　リミットチェック

解答解説

データの誤りチェックのリミットチェックに関する問題である。

アの照合チェックは、入力された得意先コードや商品コードが実在するかどうかを得意先マスタファイルや商品マスタファイルと照合してチェックすることである。

イの重複チェックは、許されない重複データがあるかどうかをチェックすることである。

ウのフォーマットチェックは、データの各項目の桁位置、けた数が決められた形式で入力されているかどうかをチェックすることである。

エのリミットチェックは、限度検査、範囲検査のことで、限度検査はデータの値が指定された値を超えていないかどうかを検査することであり、範囲検査は指定された範囲内にあるかどうかを検査する方法である。求める答えはエである。

例題演習

ニューメリックチェックの説明として，適切なものはどれか。

ア　一定の規則に従ってデータから検査文字を算出し，付加されている検査文字と比較することによって，入力データに誤りがないかどうかをチェックする。
イ　数値として扱う必要のあるデータに，数値として扱えない文字のようなものが含まれていないかどうかをチェックする。
ウ　販売数と在庫数と仕入数の関係など，関連のある項目の値に矛盾がないかどうかをチェックする。
エ　マスタファイル作成時の入力データ中に，キーの値が同じレコードが複数件含まれていないかどうかをチェックする。

解答解説

入力データのチェックに関する問題である。

ニューメリックチェックは、計算に使用する項目に数字が入っているかをチェックする。数字以外の文字が入っていると計算するときにエラーとなる。

アはチェックディジット法、イはニューメリックチェック、ウは論理チェック、エは照合チェックである。求める答えはイとなる。

txt040213 モジュール設計

① モジュールの設計とは

ⓐ 良いプログラム設計の要点

㋐ 複雑さの減少
㋑ 分割
㋒ 独立性
㋓ 階層構造化

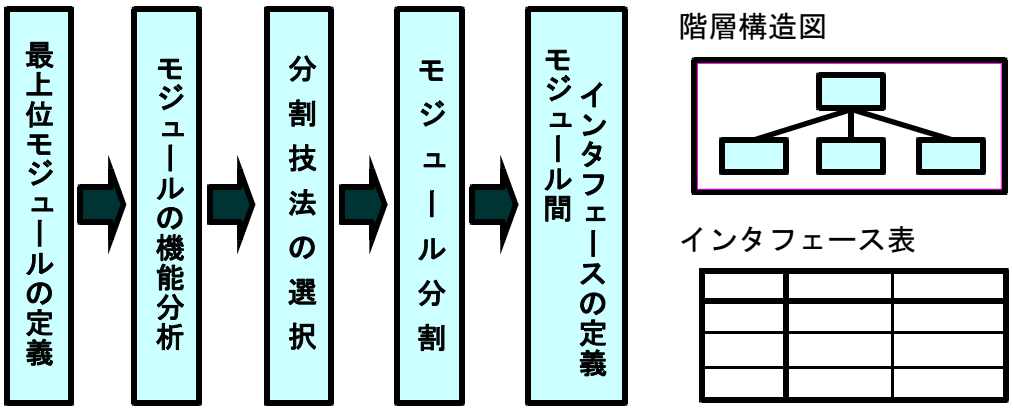
ⓑ プログラムの構造化設計

プログラム設計はプログラムをモジュールに分割し、階層構造化し、モジュール間のインタフェースを明確にし、階層構造図に表すことである。

プログラム構造化設計のポイントを次に示す。

㋐ プログラムを適当な大きさのモジュールに分割する。
㋑ 分割したモジュールを階層構造にする。
㋒ モジュール間のインタフェースを明確にする。

ⓒ 構造化設計の手順



㋐ 最上位モジュールの定義

全体的な機能を定義する。カウンタ類の初期化、ファイルのオープン・クローズ、各機能の制御を行う。

㋑ モジュールの機能分析

上位モジュールで定義した機能を分析し、それを実行するために必要な下位の機能を明ら

かにする。機能の分割や統合を検討する。入力機能、入力データチェック機能、エラーデータの処理機能、ファイルの読込・書込機能、出力機能などを行う。

㋒　分割技法の選択

モジュールを分割するための最適な分割技法を選択する。分割技法しては、ＳＴＳ分割技法、ＴＲ分割技法、共通機能分割技法、ジャクソン法、ワーニエ法などが対象になる。

㋓　モジュール分割

選択した分割技法を用いてモジュール分割を行い、階層構造図を作成する。

㋔　モジュール間インタフェースの定義

上位モジュールと直接従属モジュールのインタフェースを明確にする。

㋕　分割すべきモジュールを探す。

更に分割できるモジュールがあれば分割を繰り返す。

②　モジュール分割技法

ⓐ　分割技法とは

データの流れに着目した分割技法とデータ構造に着目した分割技法がある。データの流れに着目した分割技法に、ＳＴＳ分割、トランザクション分割、共通機能分割などがあり、データ構造に着目した分割技法に、ジャクソン法、ワーニエ法などがある。

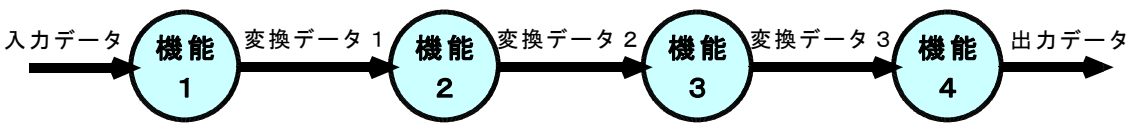
ⓑ　ＳＴＳ分割技法とは

プログラムのデータの流れに着目し、入力処理機能、変換処理機能、出力処理機能に分割していく方法である。

ⓒ　ＳＴＳ分割の手順

㋐　主要な機能を取り出し、問題構造を把握する。

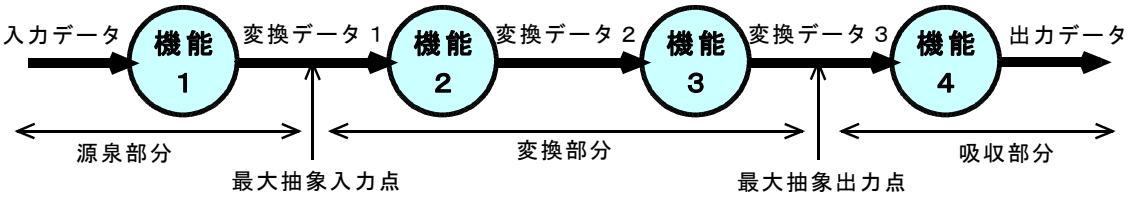
プログラム仕様書を分析し、主要なデータの流れに沿って処理機能を把握する。３～10個程度の機能を取り出し、問題の構造を明確にする。



㋑　主要なデータの流れを明確にし、機能と関連づける。

問題構造の中での、主要な入力データと出力データの流れを明確にする。ＤＦＤを用いてデータの流れを記述する。

㋒　データの変化する最大抽象入力点と最大抽象出力点を見つける。



主要なデータの流れを問題構造に沿って追いかけると、入力データが入力データの形態を無くする状態が現れる。この境界を最大抽象入力点という。主要なデータの流れを問題構造の終了点から逆に追っていくと、出力データが出力データの形態を無くする状態が現れる。この境界を最大抽象出力点という。

㋓　問題構造を最大抽象点を境に、源泉、変換、吸収の３つの主要な部分に分割し、直接従属モジュールの定義を行う。

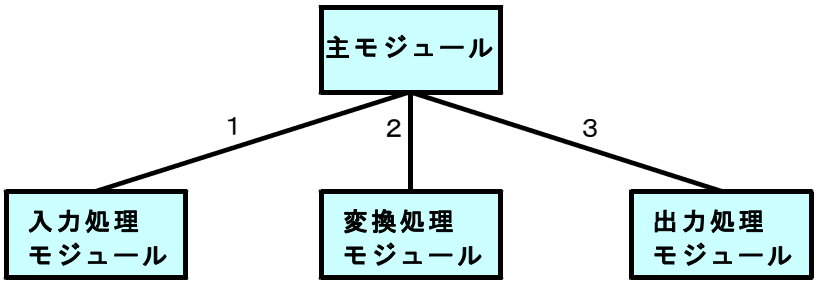
最大抽象入力点と最大抽象出力点の２点を境にして、入力側を入力処理(源泉)、出力側を出力処理(吸収)、その中間を変換処理(変換)に分け、それぞれを１つの機能としてモジュールを定義する。

㋔　上位モジュールとのインタフェースを定義する。

モジュール間の結合度が最小になるようにモジュールを分割し、階層構造図を作成する。モジュール間インタフェースを定義し、入出力するデータを明確にする。入出力インタフェース表を作成する。

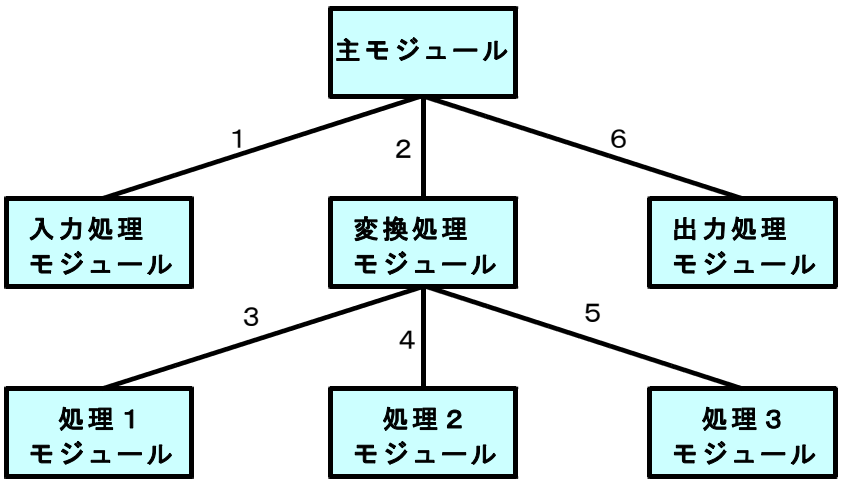
㋕　分割を繰り返す。

各モジュールを同様にして分析し、直接従属モジュールに分割する。すべてのモジュールを入力・変換・出力に分割し、インタフェースを定義する。モジュールの大きさは50～100ステップになる。



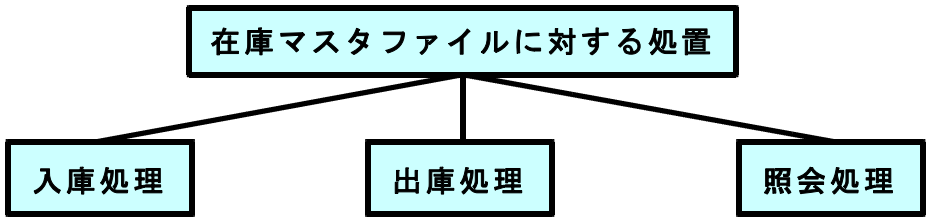
ⓓ　トランザクション（ＴＲ）分割技法とは

プログラムのトランザクションの種類に応じて、異なる処理機能をその機能単位にモジュールに分割していく方法である。



ⓔ　共通機能分割技法とは

共通の従属機能を取り出して、別個のモジュールとして定義する。同一のファイルに対する複数の機能をまとめる方法である。



③　ＳＴＳ分割の具体例

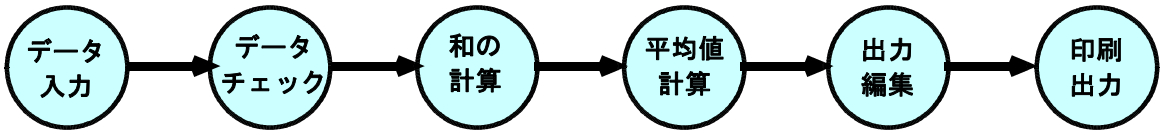
ⓐ　「データの平均値をもとめる」分割の手順

㋐　次の６つの機能に分割する。

❶　データを入力する。（データ入力）
❷　データをチェックする。（データチェック）
❸　入力データの和を求める。（和の計算）
❹　入力データの和をデータ数で割り平均値を求める。（平均値計算）
❺　結果のデータを出力様式に編集する。（出力編集）
❻　印刷出力する。（印刷出力）

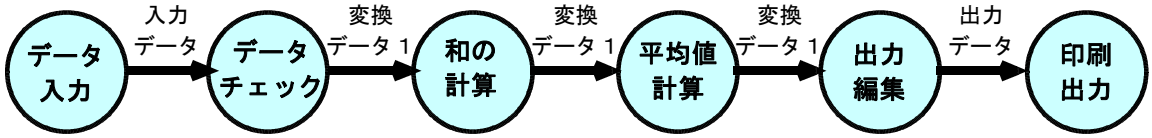
㋑　機能の再編成

主要なデータの流れとして、左から右に各機能を配置し、矢印で結ぶ。



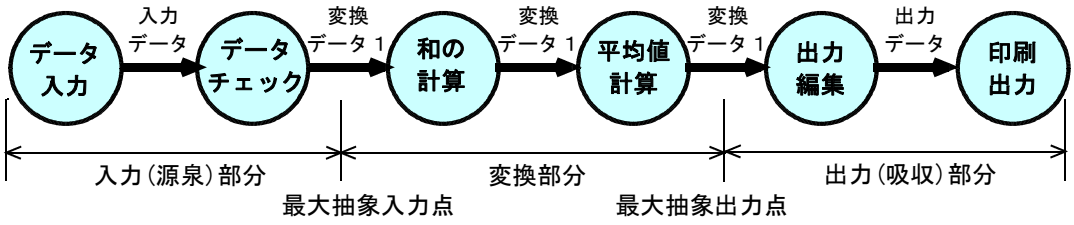
㋒　データの洗い出し

各機能間のデータに順次、入力データ、変換データ１、変換データ２、変換データ３、出力データの名前を付ける。変換データ１はチェック後のデータであり、変換データ２は入力データの和とデータの個数であり、変換データ３は平均値とその関連データである。



㋓　最大抽象点の決定

一連のデータの並びから、最大抽象入力点、最大抽象出力点を求める。最大抽象入力点は入力データが最大に抽象化される点であり、それ以降のデータはシステム内のデータとして取り扱われる。変換データ１以降はシステム内のデータとして取り扱われる。最大抽象入力点はデータをチェックすると入力データの和を求めるの間にある。最大抽象出力点は出力データが最大に抽象化される点であり、それ以前のデータはシステム内のデータとして取り扱われる。最大抽象出力点は平均値を求めると結果のデータを出力様式に編集するの間にある。



㋔　ＳＴＳに分割する。

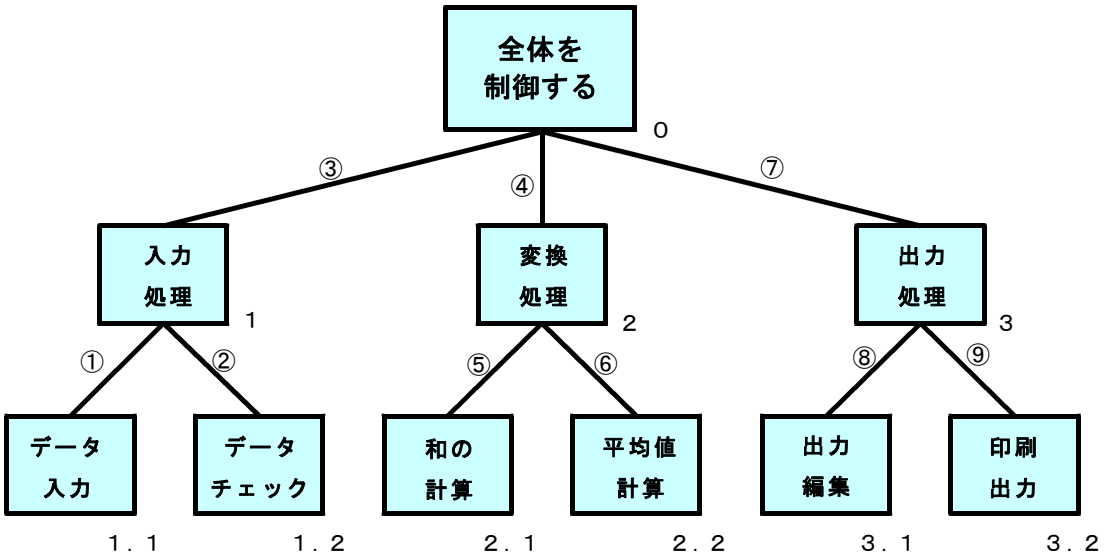
データ入力から最大抽象入力点までが源泉部分即ち入力部分になる。最大抽象入力点から最大抽象出力点までが変換部分になる。最大抽象出力点からデータ出力までが吸収部分即ち出力部分になる。

㋕　モジュール構造への変換

３つのモジュールの呼出を制御するために、全体を制御するモジュールを親モジュールとして設定する。入力部分、変換部分、出力部分を制御するモジュールを設定する。モジュール階層構造図を作成する。

㊭　モジュール間インタフェースの確定

モジュール間のインタフェース番号を設定する。データの流れをもとに各モジュール間の情報の受け渡しを決定する。インタフェース表を作成する。出力は階層構造の下位から上位に向かう方向になる。



インタフェース表

番号	入力	出力
①		入力データ
②	入力データ	変換データ１
③		変換データ１
④	変換データ１	変換データ３
⑤	変換データ１	変換データ２
⑥	変換データ２	変換データ３
⑦	変換データ３	
⑧	変換データ３	出力データ
⑨	出力データ	

④　ジャクソン法

ⓐ　ジャクソン法の図的表現法

㋐　基本

核となる構成要素であり、これ以上分割できないもので、１つのデータ項目、１つのステートメントに該当する。

㋑　連続

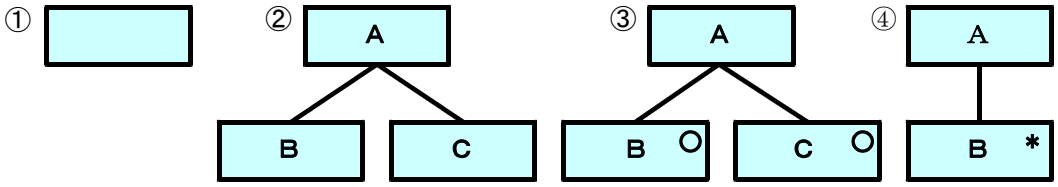
決まった順序で現れる２つ以上の相異なる基本要素で構成される構成要素で、複数のデータ項目を持つレコードに相当する。

㋒　選択

複数の部分のうち１つを選択するような構成要素で、条件によって判断を必要とする処理である。

㋓　繰返

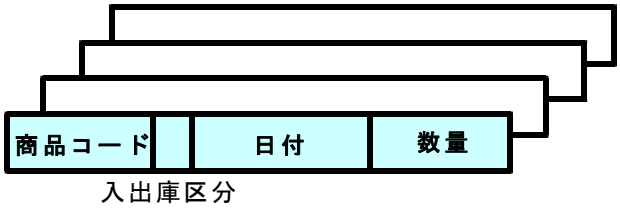
繰返しは同一の構成要素が繰り返し現れる構成要素で、配列や順次ファイルのレコードに相当する。



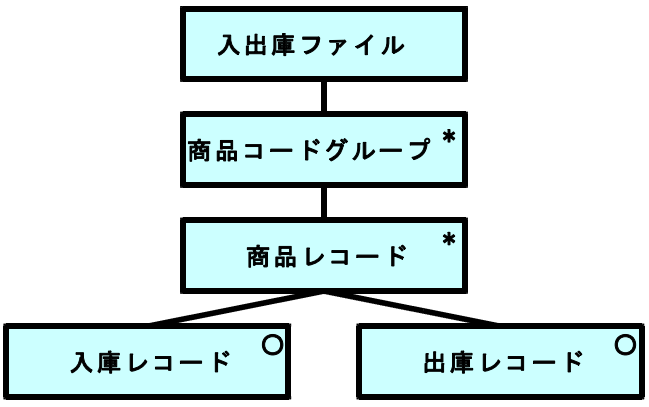
ⓑ　ジャクソン法の手順

㋐　データ構造の分析

入出庫ファイルの構造



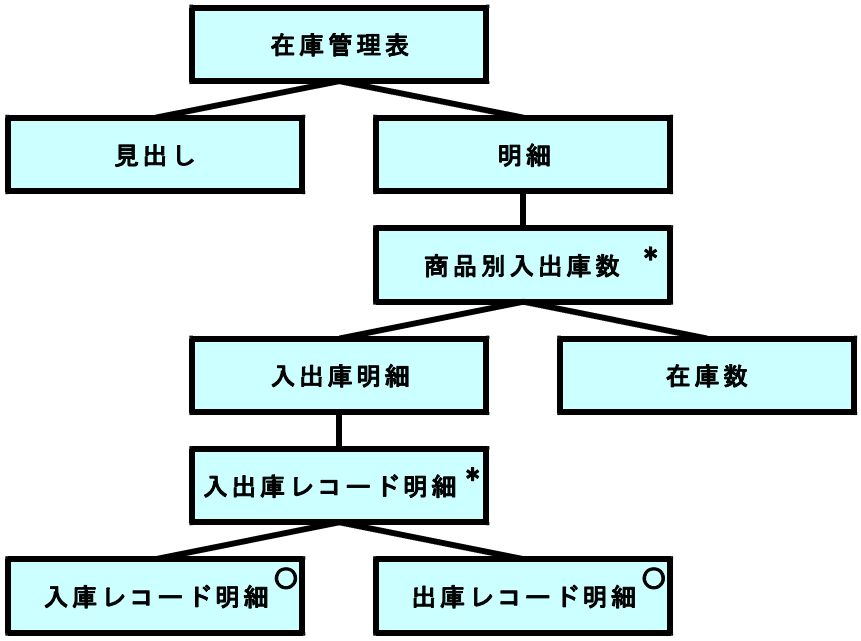
入出庫ファイルのデータ構造



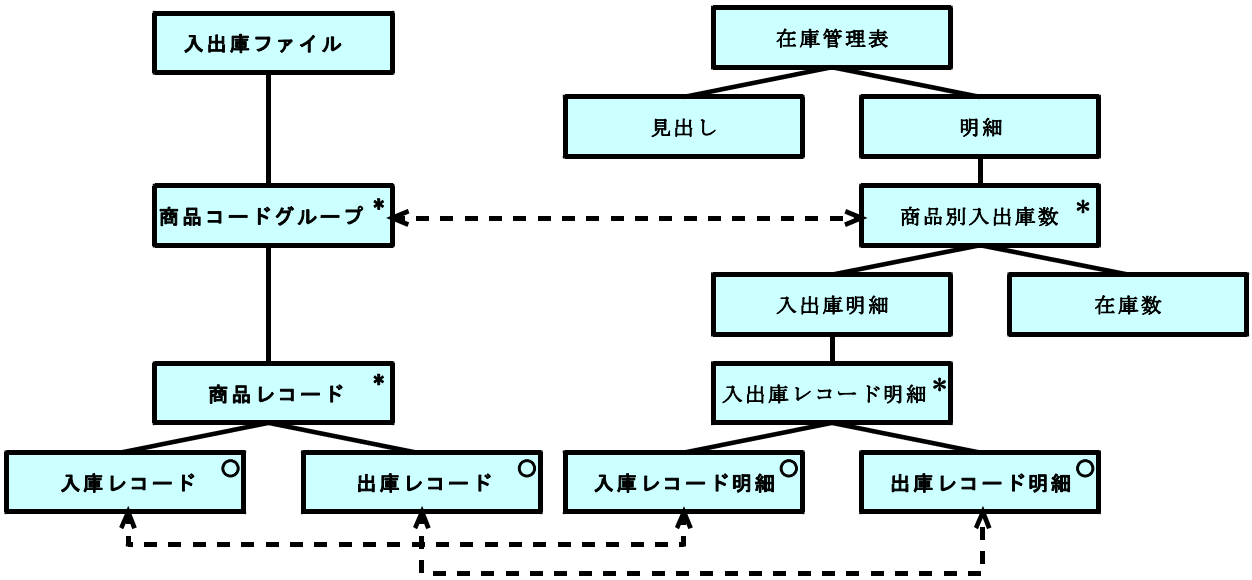
処理対象の入力と出力のデータを分析して、それぞれのデータ構造を明確にし、データ構造図を作成する

㋑　プログラム構造の作成

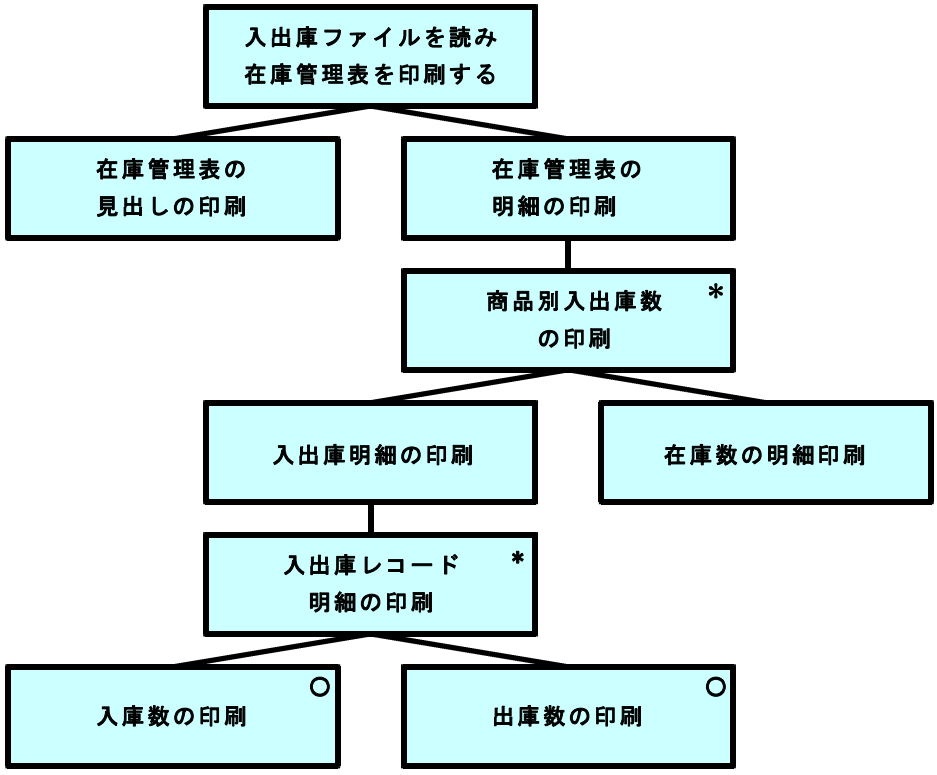
在庫管理表のデータ構造



定義した入力と出力のデータ構造の構成要素間の一対一の対応を探し、図式表示する。プログラムは入力から出力への変換であるから、一対一の対応を探すことによって、プログラムの機能を定義できる。図に示すように、商品コードグループに商品別入出庫数が対応する。入庫レコードに入庫レコード明細が対応する。出庫レコードに出庫レコード明細が対応する。商品コードグループの入力データから、商品別在庫数の結果を含む商品別入出庫数の出力データに変換するプログラム機能が必要になる。そのプログラム機能は、入庫レコードの入力データが入庫レコード明細の出力データに、出庫レコードの入力データから出庫レコード明細の出力データに変換する機能を含んだものとして構成される。



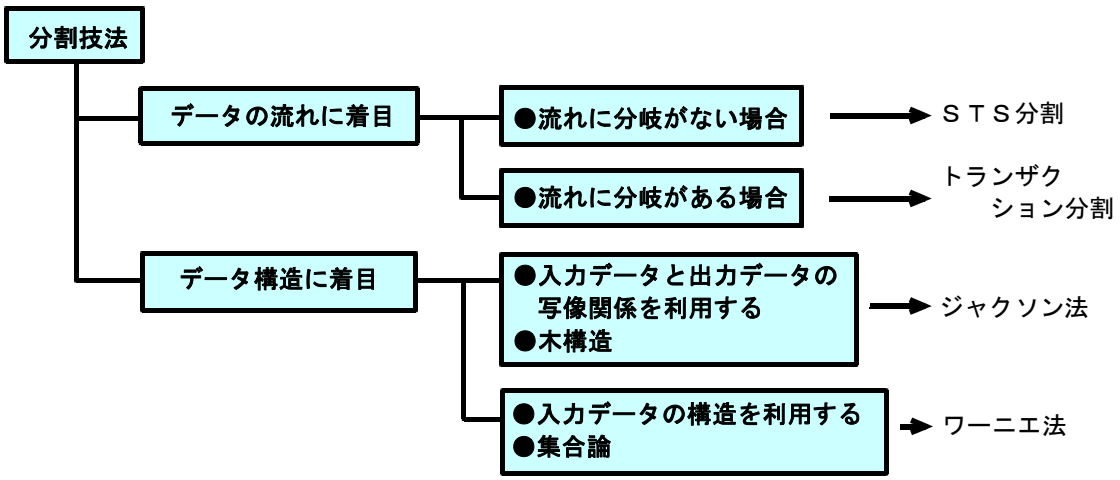
㋒　作業の定義と命令の割り当て



プログラム構造図を作成する。プログラム構造図は出力データ構造図に対応したものである。出力データ構造の繰り返し部分を作り出すにはプログラムに繰り返し構造が必要である。出力データ構造の選択の部分を作り出すにはプログラムに選択の構造が必要である。プログラム構造を表現する構成要素はデータ構造を表現する構成要素を使用する。プログラム構造ができるとプログラム構造の各構成要素に基本的な命令を割り当てる。入出力のデータ構造が複雑な場合、中間のデータ構造を作成し一対一の対応を見出す必要がある。

⑤　分割技法の選択法

ⓐ　分割技法の選択基準の流れ



⑥　モジュールの独立性

ⓐ　モジュールの特徴

㋐　複数のステートメント群が語彙としてまとまっていて、境界識別子で区切られ、単独の名前が付いている。

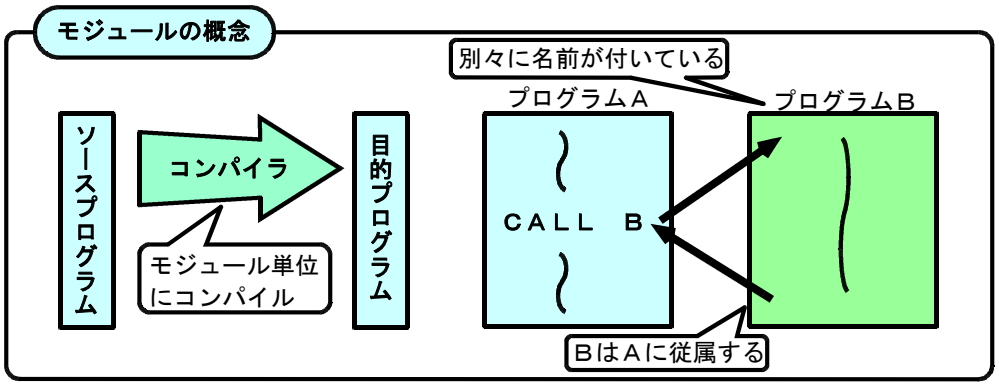
㋑　コンパイルが別々にできる。

㋒　その他のモジュールで呼び出し可能である。

㋓　呼び出したモジュールの実行は、呼び出されたモジュールの実行が終わるまで待たされる。

㋔　呼び出したモジュールの実行が終わると、実行は呼び出したモジュールに再び戻る。

ⓑ　モジュールの独立性を高める考え方



㋐　モジュール内部での関連性を最大になるようにする。

個々のモジュール内部の関連性を表す概念をモジュール強度という。モジュール強度を最大にする

㋑　モジュール間の関連性が最小になるようにする。

モジュール間の関連性を表現する概念をモジュール結合度という。モジュール結合度を最小にする。

⑦　モジュールの強度

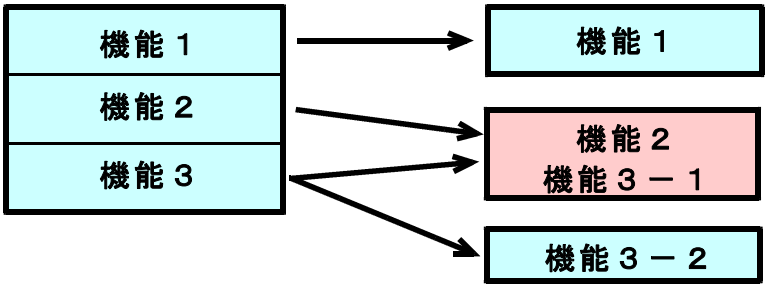
ⓐ　モジュール強度とは

モジュール強度の大きいモジュールほど独立性の高いモジュールである。従って、モジュール強度の大きいモジュールが望ましいモジュールである。モジュールの強度は、機能的強度、情報的強度、連絡的強度、手順的強度、時間的強度、論理的強度、暗号的強度の順に強度が大きい。モジュール強度が最も大きいのは機能的強度である。

ⓑ　モジュール強度の内容

㋐　暗号的強度

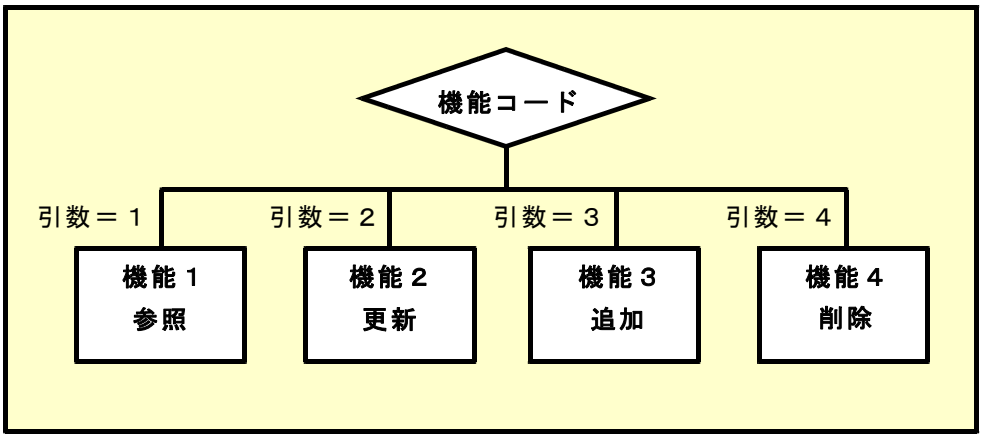
モジュール内の要素間に特別の関係が認められないモジュール化である。主記憶域の制限のために、既存のモジュールを単純に複数個に分割した場合や重複したステートメント・パターンを統合してモジュール化した場合が該当する。そのモジュールの機能を定義することができない。複数の全く関係のない機能あるいは機能の一部を実行する。モジュール内の各要素は、自モジュール内の他の要素との関連性が弱い反面、他のモジュール内の要素との強い関連を持つ傾向にある。他のモジュールの変更にも大きな影響を受けることが多く、保守がやりにくい。特定の機能が定義できないということは、再使用できる可能性が小さい。



㋑　論理的強度

ある機能を論理的にとらえてモジュール化したものである。入出力操作をまとめてすべて扱うモジュールを作る場合やすべてのデータを編集するモジュールをつくる場合が該当する。モジュール内の論理は、条件によって実行パスが異なる。関連したいくつかの機能を含み、

そのうちの１つが呼び出したモジュールによって識別され、実行される。機能的には共通するものなので、一部の論理は共有できる。モジュールが呼び出されたときモジュール内のすべてのステートメントが実行されるわけではなく、それだけ強度は弱くなる。呼び出すモジュールとのインタフェースにパラメータの取り扱いが必要になり、ミスを誘いやすくなる。特定のデータ処理を１つのモジュールに局所化できる効果はある。



㋒　時間的強度

複数の逐次的な機能を実行するが、実行する機能間に強い関連性がない。初期設定モジュールや終了処理モジュールが該当する。テーブルの初期処理は該当モジュールで実行するが、テーブルの実際の処理機能は、他のモジュールで行う。同じテーブルを介して、他のモジュールとの強い関連性を持つことになる。テーブルの変更が発生したとき、複数のモジュールが影響を受ける。

㋓　手順的強度

問題を処理するために関係している複数個の機能のうちのいくつかを実行するようなモジュールである。複数機能の中の１つだけを実行する場合は、選択機能が必要になり、論理的強度に似たものになる。大きな機能の一部の手順をモジュール化した場合、フロー・チャートの一部をモジュール化した場合が該当する。

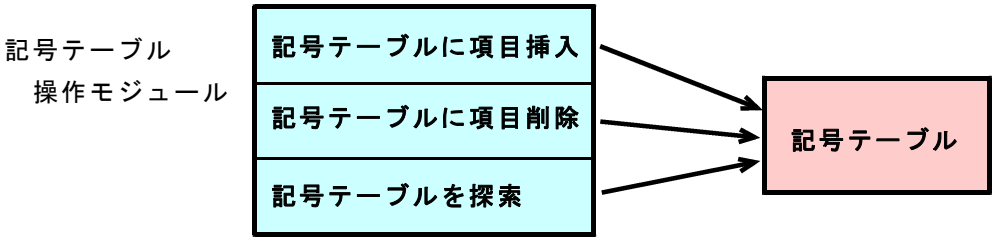
㋔　連絡的強度

手順的強度の性質をもち、その上モジュール内要素間でデータの受け渡しがあったり、同じデータを参照しているようなモジュールである。モジュール内要素がデータに関してつながっているだけ手順強度より強度は強い。

㋕　情報的強度

特定のデータ構造を扱う複数の機能を一つのモジュールにまとめたものである。同一データ構造は、特定のモジュールだけでアクセスしようという発想である。データの修正時の影響をこのモジュール内に限定することができる。データの機密保護の面からも有利である。論理的強度モジュールは１つの入り口点を持ち、実行機能の選択はパラメータを利用してい

る。情報的強度モジュールは複数個の入り口点を持ち、各入り口点は単一の固有の機能を実行する。



㋖　機能的強度

モジュール内のすべての要素が、単一機能を実行するために関連しあっている。強度としては最も強い。機能的強度を持ったモジュールの変更は、そのモジュールだけで処理でき、他のモジュールへの影響は他の強度より小さい。

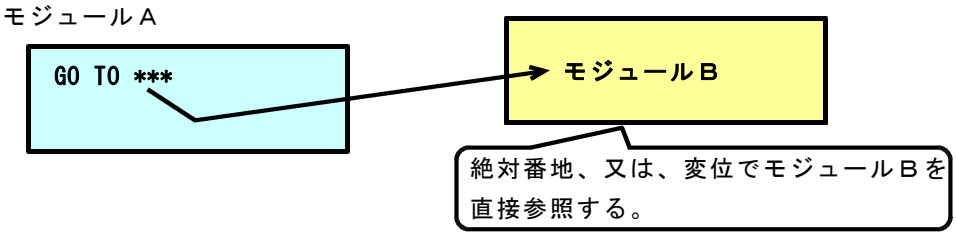
⑧　モジュールの結合度

ⓐ　モジュール結合度とは

モジュール結合度の小さいモジュールほど独立性の高いモジュールである。従って、モジュール結合度の小さいモジュールが望ましいモジュールである。モジュールの結合度は、データ結合、スタンプ結合、制御結合、外部結合、共通結合、内容結合の順に、結合度は小さい。データ結合がモジュール結合度が最も小さいモジュールである。

ⓑ　モジュール結合度の内容

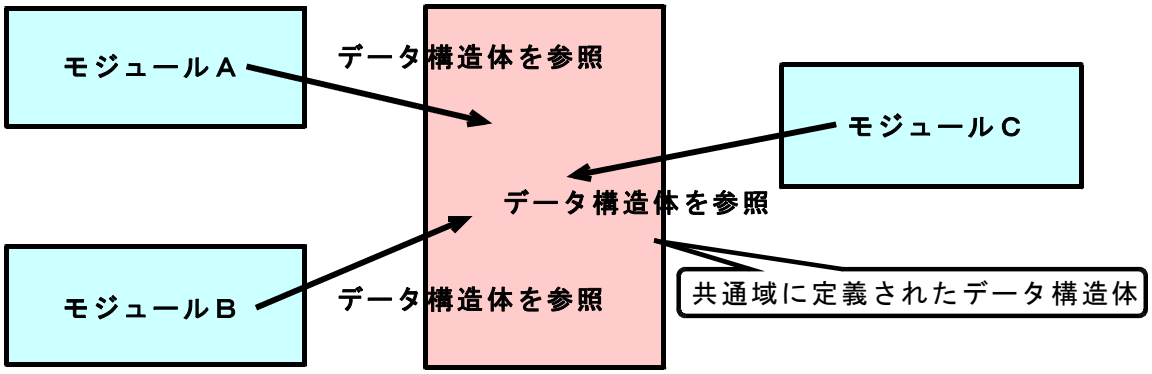
㋐　内容結合



あるモジュールと他のモジュールが一部を共有するモジュールの結合の仕方である。他のモジュール内の外部宣言していないデータを直接参照する場合や命令の一部を共有する場合が該当する。アセンブラ言語使用のモジュールに発生する。一方のモジュールの変更が他のモジュールに影響を及ぼすことになる。他のモジュールのことまで考えて変更する必要がある。

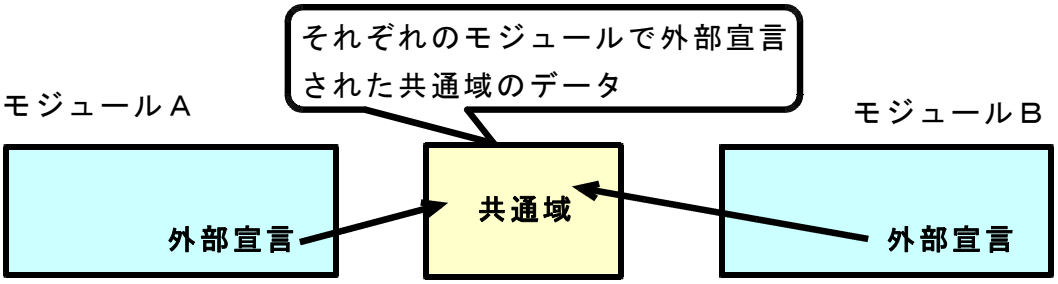
㋑　共通結合

共通域に定義したデータ構造体を複数のモジュールが共有するような結合の方法である。共通域の一部のデータ構造体の修正により、共通域参照の全モジュールのコンパイルのし直しが必要になったり、共通域のデータは、もともとそのデータに無関係なモジュールでもその気になれば使えるため、結合度が高いモジュールといえる。共通結合しているモジュールを他のモジュールが使用したいとき、使いにくい。また、共通域のデータはモジュール間のインタフェース上に現れないので、ソースコードによるモジュール機能の解読を難しくする。モジュールＡの都合でデータＸの長さを変更したとき、本来その変更に無関係なモジュールＢ、Ｃも再コンパイルの対象になる。モジュールＡの設計者がデータＸがモジュールＣで使用されていることを知らない場合には思わぬトラブルが発生する可能性がある。



㋒　外部結合

外部宣言したデータを共有する場合である。共有結合とよく似ているが、必要なデータだけを外部宣言するので不必要なデータまで共有することがないのでそれだけ結合度が弱くなる。



㋓　制御結合

呼び出すモジュールが、呼び出されるモジュールの制御を指示するデータをパラメータとして渡すやり方である。スイッチとかインデックスをパラメータとして渡す。呼び出し側は呼び出されるモジュールの論理を知っている必要がある。相手をブラックボックス化できないだけ、結合度が強くなる。呼び出されるモジュールの強度は論理的強度になる。データの共有といったことがないので、共通結合や外部結合よりも結合度は弱くなる。

㊔　スタンプ結合

共通域にないデータ構造を２つのモジュールで受け渡しするような場合である。データ構造を受け渡ししながら、構造内の一部データを使用しない場合である。不必要なデータまで受け渡しする点がデータ結合よりも結合度が強くなる。データ構造を受け渡ししても、構造のすべてのデータが処理されるならば、データ結合と見なせる。

㊕　データ結合

モジュール間のインタフェースとして、スカラ型のデータ要素だけをパラメータとして受け渡す。相手のモジュールをブラックボックスとして扱うことが可能となり、結合度が一番弱くなる。複合設計では結合度の一番弱いデータ結合を良しとしている。モジュール間の結合は単純に明確化されたパラメータでデータを受け渡しする型が一番良いのである。モジュールＡは入力としてＸをモジュールＢに渡し、出力としてＹをもらうことがわかっておればそれで良いのである。モジュールＢがＸをどのようにしてＹに変換するのか、その手順はＡには必要ないことである。モジュールＢの手順変更によってモジュールＡが影響を受けることはない。モジュールＡがモジュールＢのデータを直接参照する内容結合のようなときは、モジュールＢの変更がモジュールＡにそのまま影響を及ぼすことになる。このような場合は保守がやりにくく、トラブル発生の原因になる。プログラムのモジュール化をはかる場合、モジュールを機能的にまとめ、モジュール間のインタフェースはデータ結合で行うようにすれば、モジュールの独立性が高くなり、信頼性の高いプログラムが設計できることになる。

⑨　ソフトウェア設計と複合設計

ⓐ　ソフトウェア設計とは

ソフトウェアの設計は、要求されたコンピュータシステムに対して、ソフトウェアを実装するために、段階的に具体化していく作業のことである。ソフトウェアの要求を、プログラム構造とデータ構造に変換していく工程であり、分割と統合という考え方を利用して、ソフトウェアの機能を階層的にいくつかの機能に分割していくと、階層化されたある機能は、いくつかの独立した小さな機能単位から構成されるようになる。

ソフトウェアを階層的に構成する個々の機能は、システム、サブシステム、プログラム、モジュールに相当する。システムからモジュールへと段階的に詳細化していくプロセスがソフトウェア設計に相当する。

ⓑ　複合設計

プロセスを設計する技法に、処理するプロセスに着目しながら進める考え方とデータに着目しながら進める考え方、データとプロセスを一体化して進める考え方がある。それぞれ、プロセス指向設計、データ指向設計、オブジェクト指向設計という。

複合設計は、構造化設計技法を取り入れたもので、モジュール構造とそのインタフェースを設計する段階で、モジュール強度を最大に、モジュール強度を最小にすることによって、モジ

ュールの独立性を高める考え方である。モジュールの独立性が高いと言うことは、個々のモジュールが独自の機能だけで実現しているとともに、他のモジュールへの波及効果が少ないことになる。これによって、保守性を高めることが可能になる。具体的には、構造化設計技法で使用されるＳＴＳ分割法、ＴＲ分割法が用いられる。

ⓒ　複合設計の手順

㋐　問題の構造の概観を記述する。

データの流れを基本にして、３～10個の機能で問題の記述を行う。

㋑　問題の中の主要な入力データと出力データの流れを明らかにする。

㋒　主要な入力データの流れを問題の構造に沿ってトレースする。

入力データは最初はっきり入力の形をとっているが、徐々に形を変え、そのうちに、最早入力データと言えなくなる点に到達する。これが最大抽象入力点である。

㋓　同様の分析を主要出力データの流れについて行う。

最大抽象出力点を求める。

㋔　最大抽象入力点と最大抽象出力点を基準に、問題構造を三つの主要部分に分割する。

源泉部分、中央変換部分、吸収部分に分かれる。

㋕　モジュール間のインタフェースを定義する。

インタフェースはデータ結合になるように配慮する。

㋖　更に、下位のモジュールについても同様な分析をする。

ⓓ　独立性の評価基準

〔独立性〕

モジュール強度		低	モジュール結合度	
暗合的強度	弱		強	内容結合
論理的強度				共通結合
時間的強度				外部結合
手順的強度				制御結合
連絡的強度				スタンプ結合
情報的強度				データ結合
機能的強度	強	高	高	非直接結合

例題演習

プログラムのモジュール化に関する記述として、適切なものはどれか。

ア　異なったモジュール間の情報の受け渡しは、ファイルを仲介にして行う。
イ　主記憶装置の容量を許す限り、モジュールを大きくし、その個数を少なくすべきである。
ウ　プログラミングは容易になるが、保守は難しくなる。
エ　モジュール単位にコーディング、デバッグが可能になるので、プログラム開発、テストの生産性向上に役立つ。

解答解説

モジュール化に関する問題である。

アのモジュール間の情報の受け渡しは、データで行う。ファイルで行うのはプロセスフローにおける処理プログラム間の場合である。

イのモジュールの大きさは、可能な限り小さくしモジュール間の結合度を弱くするのが適切な処理であり、大きくして、個数を少なくするのはモジュールの独立性から問題がある。

ウの適切なモジュール化は、モジュール強度が大きく、モジュール結合度が弱くなるため、モジュールの独立性が高くなり、保守を容易にする。

エのモジュール単位のコーディング、デバッグが可能になり、開発やテストの生産性が向上するは適切な記述である。求める答えはエである。

例題演習

モジュール分割技法の中で，データの流れに沿って，入力処理機能，変換機能，出力処理機能へと分割する技法はどれか。

ア　ＳＴＳ分割　　イ　共通機能分割
ウ　ジャクソン法　　エ　トランザクション分割

解答解説

モジュール分割技法のＳＴＳ分割に関する問題である。

アのＳＴＳ分割は、プログラムを入力データ処理、入力データから出力データへの変換処理、出力データ処理の３つのモジュールに分割する手法である。求める答えはアである。

イの共通機能分割は、共通的な機能を取り出して、複数のモジュールで実行できる独立したモジュールとして定義する方法である。

ウのジャクソン法は、データ構造からプログラム構造を決めるデータ中心の構造化技法で、基本、連続、選択、繰り返しの４つの基本図式を用いるジャクソン図を使って、データ構造とプログラム構造を階層構造で表す。

エのトランザクション分割は、オンライントランザクション処理などのプログラムで、トランザクションの種類ごとにモジュールを分割する手法である。

例題演習

次に示すプログラム構造化設計の手順のうち、正しい手順はどれか。

Ａ　分割技法の選択　　Ｂ　インタフェースの定義
Ｃ　最上位モジュールの定義　　Ｄ　モジュールの機能分析
Ｅ　モジュール分割　　Ｆ　分割すべき他のモジュールの検討

ア　Ａ→Ｄ→Ｃ→Ｅ→Ｂ→Ｆ　　イ　Ａ→Ｃ→Ｄ→Ｅ→Ｂ→Ｆ
ウ　Ｃ→Ｄ→Ａ→Ｅ→Ｂ→Ｆ　　エ　Ｃ→Ａ→Ｄ→Ｅ→Ｂ→Ｆ

解答解説

プログラムの構造化設計の手順に関する問題である。

プログラム構造化設計の手順は、最上位モジュールの定義（Ｃ）、モジュールの機能分析（Ｄ）、分割技法の選択（Ａ）、モジュールの分割（Ｅ）、インタフェースの定義（Ｂ）、分割すべき他のモジュールの検討（Ｆ）の順に行う。Ｃ→Ｄ→Ａ→Ｅ→Ｂ→Ｆの順になり、求める答えはウとなる。

例題演習

“データを読み込み、数字データだけを選んでその平均値を表示する”プログラムをＳＴＳ分割したとき、それぞれの機能は吸収、源泉、変換のどの部分に分類されるか。

	機　　能			
	データ入力	数字選択	平均値算出	表示
ア	吸収	吸収	源泉	変換
イ	吸収	源泉	源泉	変換
ウ	源泉	源泉	変換	吸収
エ	源泉	変換	変換	吸収

解答解説

平均値を求めるプログラムのＳＴＳ分割に関する問題である。

源泉は入力処理部分、吸収は出力処理部分であるから、データ入力は源泉、表示は吸収になる。平均値算出は変換に相当する。数字選択は入力のデータ構造で実行されるものであり、源泉に相当する。従って、源泉・源泉・変換・吸収になる。求める答えはウとなる。

例題演習

オンラインリアルタイム処理のように、入力トランザクションの種類に応じて処理が異なる場合に有効な分割技法はどれか。

ア　ＳＴＳ分割　　イ　ジャクソン法　　ウ　共通機能分割　　エ　ＴＲ分割

解答解説

トランザクション分割に関する問題である。

アのＳＴＳ分割は、プログラムを入力データ処理、入力データから出力データへの変換処理、出力データ処理の３つのモジュールに分割する手法である。

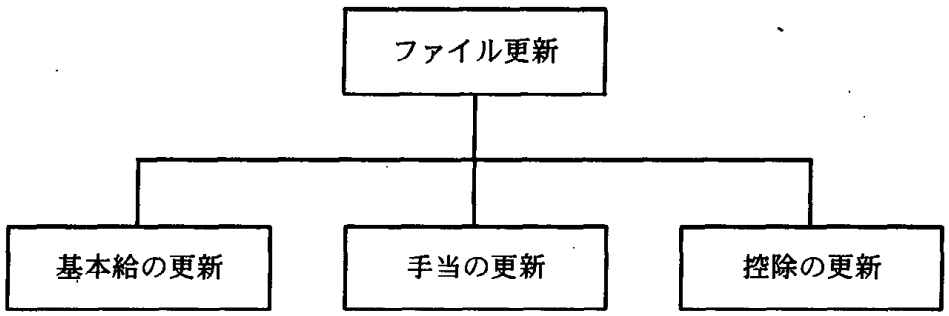
イのジャクソン法は、データ構造からプログラム構造を決めるデータ中心の構造化技法で、基本、連続、選択、繰り返しの４つの基本図式を用いるジャクソン図を使って、データ構造とプログラム構造を階層構造で表す。

ウの共通機能分割は、個々のプログラムで共通する処理を、独立したモジュールとするモジュール分割法である。

エのトランザクション分割は、データの流れに着目した分割技法で、データの流れに分岐がある場合に用いる。入力トランザクションの種類に応じて処理が異なる場合の手法であるから、トランザクション分割であり、求める答えはエとなる。

例題演習

基本給の更新，手当の更新，控除の更新に関する伝票を個別に受け付け，給与計算用のファイルを更新するプログラムを，図のようにモジュール分割した。このモジュール分割の方法の名称はどれか。



ア　ＳＴＳ分割法　　イ　ジャクソン法

ウ　トランザクション分割法　　エ　ワーニエ法

解答解説

トランザクション分割に関する問題である。

アのＳＴＳ分割は、プログラムを入力データ処理、入力データから出力データへの変換処理、出力データ処理の３つのモジュールに分割する手法である。

イのジャクソン法は、データ構造からプログラム構造を決めるデータ中心の構造化技法で、基本、連続、選択、繰り返しの４つの基本図式を用いるジャクソン図を使って、データ構造とプログラム構造を階層構造で表す。

ウのトランザクション分割は、オンライントランザクション処理などのプログラムで、トランザクションの種類ごとにモジュールを分割する手法である。

エのワーニエ法は、入出力データを集合としてとらえる事務処理向きの構造化設計手法で、順次、選択、繰り返しでデータ構造を表現する。

階層構造図では、基本給の更新、手当の更新、控除の更新とモジュールが処理機能別に分けられている。これはトランザクション分割である。求める答えはウとなる。

例題演習

構造化分析で作成したデータフロー図から、構造化設計における構造化チャートへの変換する技法はどれか。

ア　ＫＪ法　　イ　ＯＭＴ法
ウ　ジャクソン法　　エ　トランザクション分割法

解答解説

構造化分析手法のトランザクション分割に関する問題である。

アのＫＪ法は、断片的な情報を整理するための手法で、名刺大のカードに発言を要約し、カードの内容の親近性によって分類し、図解し、文章化する手順で情報の整理を行う手法である。

イのＯＭＴ法は、オブジェクト指向を使ってシステム分析／設計開発方法論の一つである。システムをオブジェクトモデル、動的モデル、機能モデルを使って表現する。

ウのジャクソン法は、データ構造からプログラム構造を決めるデータ中心の構造化技法で、基本、連続、選択、繰り返しの４つの基本図式を用いるジャクソン図を使って、データ構造とプログラム構造を階層構造で表す。

エのトランザクション分割は、データの流れに着目した分割技法で、データの流れに分岐がある場合に用いる。トランザクションを入力するモジュール、トランザクションの属性ごとに振り分けるモジュール、それぞれのトランザクションごとに変換するモジュールに分ける。

ＤＦＤを利用して、階層構造図の構造化チャートに変換する技法はトランザクション分割であり、求める答えはエとなる。

例題演習

データ構造に着目したモジュール分割技法はどれか。

ア　共通機能分割　　イ　源泉／変換／吸収分割（ＳＴＳ分割）
ウ　ジャクソン法　　エ　トランザクション分割（ＴＲ分割）

解答解説

データ構造に着目したモジュール分割技法に関する問題である。

アの共通機能分割は、データの流れに着目した分割技法で、個々のプログラムで共通する処理を、独立したモジュールとするモジュール分割法である。

イのＳＴＳ分割は、データの流れに着目した分割技法で、プログラムを入力データ処理、入力データから出力データへの変換処理、出力データ処理の３つのモジュールに分割する手法である。

ウのジャクソン法は、データの構造に着目して、入力データ、出力データの構造、構成要素のデータ項目に対応した形で分割を行う。求める答えはウとなる。

エのトランザクション分割は、データの流れに着目した分割技法で、データの流れに分岐がある場合に用いる。

例題演習

プログラムの構造化設計におけるワーニエ法の説明として、適切なものはどれか。

ア　扱うデータの構造に着目して、入出力データの構造図を作成し、次に入力データの構造図をもとにプログラム構造図を作成する方法である。
イ　扱うデータの流れに着目し、データフロー上の機能を源泉(source)、変換(transform)、吸収(sink) にグループ分けする方法である。
ウ　ソフトウェアをデータとプロセスの集合としてとらえ、一本化することによってモジュールとしての独立性を高める方法である。
エ　プログラムの制御構造に着目し、呼出し関係を表す制御フローに基づいて論理設計を行う方法である。

解答解説

ワーニエ法に関する問題である。

ワーニエ法は、１９７０年代の初めにフランスのワーニエによって確立された構造的プログラミング技法である。主として事務処理分野での利用を目指している。ファイルやデータの構造の分析を行い、これをもとにプログラムの構造的分析であるワーニエ図を作成する。作成手順は、問題の分析、データ構造の分析、データ構造からプログラム構造への変換、ワーニエ図からプログラム流れ図への手順で進める。データ分析では、入力データの構造と出力データの構造を集合論を利用して分析する。ファイルを一つの集合としてとらえ、部分集合に細分化していく。

アはワーニエ法、イはＳＴＳ分割法、ウはオブジェクト指向分析設計法、エは構造化チャートを利用した構造化手法である。求める答えはアとなる。

例題演習

システム設計およびプログラム構造の作成に集合論を適用しているのを特徴としているのはどれか。

ア　ジャクソン法　　イ　ワーニエ法
ウ　段階的詳細化技法　　エ　ブラックボックス法

解答解説

データ構造に着目したワーニエ法に関する問題である。

アのジャクソン法は、データ構造からプログラム構造を決めるデータ中心の構造化技法で、基本、連続、選択、繰り返しの４つの基本図式を用いるジャクソン図を使って、データ構造とプログラム構造を階層構造で表す。木構造を利用した分割技法である。

イのワーニエ法は、入出力データを集合としてとらえる事務処理向きの構造化設計手法で、

順次、選択、繰り返しでデータ構造を表現する。データ構造に着目した分割技法である。求める答えはイとなる。

ウの段階的詳細化は、上位から下位へ段階的に大きな機能を小さい機能に分割しながら設計を進めていく方法である。

エのブラックボックス法は、モジュールをブラックボックスと見なし、機能仕様に基づき作成したテストデータでテストを行う手法である。

例題演習

プログラムのモジュール化設計においてモジュールの分割を行う際には、それぞれの独立性に留意することが大切である。モジュールの独立性を評価するための尺度に関する概念はどれか。

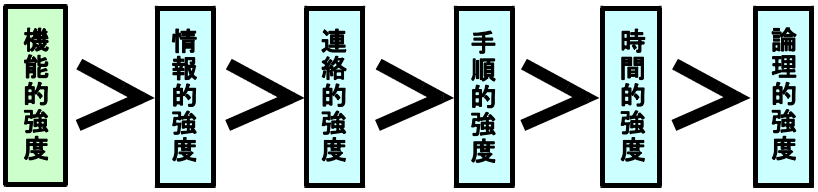
ア　モジュール間結合度　　イ　モジュール深度
ウ　モジュール精度　　エ　モジュール難易度

解答解説

モジュールの独立性の評価に関する問題である。

モジュール強度は、モジュール強度は独立性を高めるための要素の１つで、強度の高いほど独立性が高いモジュールである。強度の高低の分類は、部分の機能の自己完結度や凝縮度で分けることができる。

モジュール強度は、機能的強度、情報的強度、連絡的強度、手順的強度、時間的強度、論理的強度、暗号的強度に分類され、機能的強度が最も独立性が高い。



モジュールの独立性を評価する尺度には、モジュール強度とモジュール間結合度がある。この問題ではモジュール間結合度が該当する。求める答えはアとなる。

例題演習

モジュール内部の関連性を示す尺度にモジュールの強度がある。次のうち、強度の強い順に並んでいるのはどれか。

ア　暗号的強度、情報的強度、手順的強度、連絡的強度、時間的強度
イ　機能的強度、手順的強度、情報的強度、連絡的強度、論理的強度
ウ　機能的強度、情報的強度、連絡的強度、手順的強度、時間的強度
エ　情報的強度、連絡的強度、手順的強度、時間的強度、暗号的強度

解答解説

モジュール強度に関する問題である。

モジュール内部の関連性を表すモジュール強度の種類

① 暗号的強度は、モジュール内の要素間に特別な関係が認められない場合で、既存モジュールを単純に分解したり、重複したコーディングを統合したりする場合である。

② 論理的強度は、関連したいくつかの機能を含み、そのうちの一つが別のモジュールによって選択される。モジュール内のすべてのステートメントが実行されるわけではない。

③ 時間的強度は、複数の逐次的な機能を実行するが、実行する機能間にはあまり強い関連性がない。

④ 手順的強度は、問題を処理するために関係している複数個の機能のうちいくつかを実行するようなモジュールである。

⑤ 連絡的強度は、手順的強度の性質を持ち、その上、モジュール内要素間でデータの受け渡しがあったり、同じデータを参照したりするモジュールである。

⑥ 情報的強度は、特定のデータ構造を扱う複数の機能を一つのモジュールにまとめたもの。

⑦ 機能的強度は、すべての要素が一つの機能を実行するために関連し合っているモジュールで、一番強い強度のモジュールである。

上の説明の①～⑦はモジュール強度が弱い順に並んでいる。最も強度が強いのは機能的強度である。従って、求める答えは、機能的強度、情報的強度、連絡的強度、手順的強度、時間的強度の順になる。求める答えはウとなる。

例題演習

モジュール間の関係を示す尺度のモジュールの結合度の大きい順に並んでいるのはどれか。

ア　スタンプ結合、データ結合、内容結合、外部結合、制御結合

イ　内容結合、外部結合、制御結合、スタンプ結合、データ結合

ウ　共通結合、外部結合、制御結合、データ結合、スタンプ結合

エ　データ結合、スタンプ結合、制御結合、外部結合、共通結合

解答解説

モジュール結合度に関する問題である。

モジュール間の関連性を表すモジュール結合度の種類

① 内容結合は、他のモジュール内の外部宣言していないデータを直接参照する。依存度が非常に高く、変更が他に影響しやすい。

② 共通結合は、共通域に定義したデータをいくつかのモジュールが共有するような結合の方法である。

③ 外部結合は、外部宣言したデータの共有である。必要なデータだけを外部宣言するので、不必要なデータまで共有することはない。結合度は弱くなる。

④ 制御結合は、呼び出すモジュールが、呼び出されるモジュールの制御を指示するデータをパラメータとして渡すやり方である。スイッチとかインデックスをパラメータとして渡す。データ結合に比べて結合度が強くなる。データの共有がないので外部結合より結合度

は弱い。

⑤　スタンプ結合は、共通域にないデータ構造を二つのモジュールで受け渡しする場合。不必要なデータまで受け渡しする点がデータ結合より結合度が強くなる。

⑥　データ結合は、モジュール間のインタフェースとして、スカラ型のデータ要素だけをパラメータとして受け渡す。相手のモジュールをブラックボックスとして扱うことが可能になり、結合度は一番弱い。

モジュールの結合度の大きい順に並べると、

内容結合、共通結合、外部結合、制御結合、スタンプ結合、データ結合

となり、求める答えはイとなる。

例題演習

モジュールの独立性の尺度であるモジュール結合度は，弱いほど独立性が高くなる。次のうち，モジュールの独立性が最も高い結合度をもつものはどれか。

ア　共通結合　　イ　スタンプ結合　　ウ　データ結合　　エ　内容結合

解答解説

モジュール結合度に関する問題である。

アの共通結合は、プログラムの共有領域に定義したデータを関係する各モジュールがそのデータを共有する方式である。共有するデータの構造が変更になった場合、共通領域のデータ定義部分と各モジュールのデータ定義部分の修正が必要となる。結合度が強くモジュールの独立性が高くない。

イのスタンプ結合は、モジュール間の引数としてデータ構造名を渡してモジュール間でデータを共有する。共通領域にデータ領域を定義しないので、モジュール間の関連性は弱くなる。

ウのデータ結合は、モジュール間はデータ要素のみ引数で受け渡す。モジュールを修正しても影響を及ぼす範囲が少ない。データ結合はモジュールの独立性が最も高い。

エの内容結合は、他のモジュールのある部分を直接参照する方式で、結合度が最も強い。

独立性が最も高い結合度はデータ結合で、求める答えはウとなる。

例題演習

ソフトウェアのモジュール設計において、信頼性、保守性を向上させるためのアプローチとして、望ましいものはどれか。

ア　モジュール強度を高く、結合度を高くする。
イ　モジュール強度を高く、結合度を低くする。
ウ　モジュール強度を低く、結合度を高くする。
エ　モジュール強度を低く、結合度を低くする。

解答解説

モジュール設計の独立性に関する問題であ。

モジュールの信頼性、保守性を高めるためにはモジュールの独立性を確保する必要があり、独立性を確保するためにはモジュールの強度を高くし、結合度を低くする必要がある。求める答えはイである。

例題演習

モジュール結合度が最も弱いモジュールはどれか。

ア　単一のデータ項目を大域的データで受け渡すモジュール
イ　単一のデータ項目を引数で受け渡すモジュール
ウ　データ構造を大域的データで受け渡すモジュール
エ　データ構造を引数で受け渡すモジュール

解答解説

モジュール結合度に関する問題である。

モジュール結合度の小さいモジュールほど独立性の高いモジュールである。従って、モジュール結合度の小さいモジュールが望ましいモジュールである。モジュールの結合度は、データ結合、スタンプ結合、制御結合、外部結合、共通結合、内容結合の順に、結合度は小さい。データ結合がモジュール結合度が最も小さいモジュールである。

アは外部結合、イはデータ結合、ウは共通結合、エはスタンプ結合であり、求める答えはイとなる。

例題演習

モジュール強度が最も高いものはどれか。

ア　あるデータを対象として逐次的に複数の機能を実行するモジュール
イ　異なる入力媒体からのデータを処理するモジュール
ウ　単一の機能を実行するモジュール
エ　特定の時点で必要とされる作業のすべてを含んでいるモジュール

解答解説

モジュール強度に関する問題である。

モジュール強度の大きいモジュールほど独立性の高いモジュールである。従って、モジュール強度の大きいモジュールが望ましいモジュールである。モジュールの強度は、機能的強度、情報的強度、連絡的強度、手順的強度、時間的強度、論理的強度、暗号的強度の順に強度が大きい。モジュール強度が最も大きいのは機能的強度である。

機能的強度は、モジュール内のすべての要素が、単一機能を実行するために関連しあっている。強度としては最も強い。機能的強度を持ったモジュールの変更は、そのモジュールだけで処理でき、他のモジュールへの影響は他の強度より小さい。

アは手順的強度、イは論理的強度、ウは機能的強度、エは時間的強度である。求める答えはウとなる。

例題演習

モジュールの独立性を高めるにはモジュール結合度を弱くする。モジュール間の情報の受渡しに関する記述のうち、モジュール結合度が最も弱いものはどれか。

ア　共通域に定義したデータを、関係するモジュールが参照する。
イ　制御パラメタを引数として渡し、モジュールの実行順序を制御する。
ウ　データ項目だけをモジュール間の引数として渡す。
エ　必要なデータだけを外部宣言して共有する。

解答解説

モジュール結合度に関する問題である。

結合度の弱い順は、データ結合、制御結合、外部結合、共通結合の順になる。

アは共通結合、イは制御結合、ウはデータ結合、エは外部結合である。

結合度が最も低いのはデータ結合であり、求める答えはウとなる。

例題演習

図は，五つの関数Ａ～Ｅからなるプログラムの構成を表したものである。表に，これらの関数間のインタフェースを示す。図中の番号は関数間のインタフェースを示し，表中の項目“Ｎｏ”の列と対応している。このほかに，関数Ａ，Ｄ，Ｅ は，特定のデータ領域を参照するという関係がある。関数ＡとＥの間のモジュール結合関係はどれか。

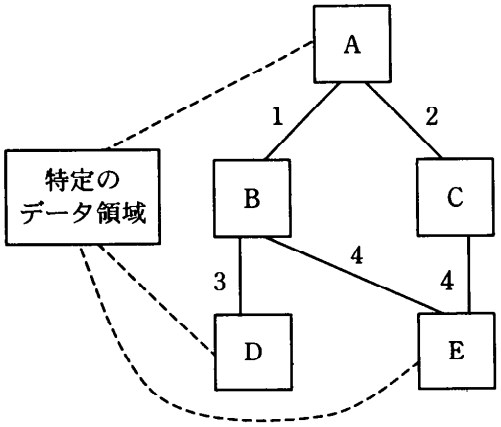


図　プログラムの構成

表　関数間のインタフェース

No	引数	戻り値
1	データ項目の内容	データ項目の内容
2	データ項目の内容	データ項目の内容
3	機能コード	なし
4	なし	リスト

ア　共通結合　　イ　データ結合
ウ　内容結合　　エ　なし(非直接結合)

解答解説

複合設計のモジュール結合に関する問題である。

アの共通結合は、共通域に定義したデータをいくつかのモジュールが共有するような結合の方法である。

イのデータ結合は、モジュール間のインタフェースとして、スカラ型のデータ要素だけをパ

ラメータとして受け渡す。

ウの内容結合は、他のモジュール内の外部宣言していないデータを直接参照する。

ＡとＥは特定のデータ領域を共に参照しているため共通結合であり、求める答えはアとなる。

例題演習

モジュール設計書を基にモジュール強度を評価した。適切な評価はどれか。

〔モジュール設計書（抜粋）〕

上位モジュールから渡される処理コードに対応した処理をする。処理コードが“Ｉ”のときは挿入処理，処理コードが“Ｕ”のときは更新処理，処理コードが“Ｄ”のときは削除処理である。

ア　これは“暗合的強度”のモジュールである。モジュール内の機能間に特別な関係はなく，むしろ他のモジュールとの強い関係性をもつ可能性が高いので，モジュール分割をやり直した方がよい。

イ　これは“情報的強度”のモジュールである。同一の情報を扱う複数の機能を，一つのモジュールにまとめている。モジュール内に各処理の入口点を設けているので，制御の結びつきがなく，これ以上のモジュール分割は不要である。

ウ　これは“連絡的強度”のモジュールである。モジュール内でデータの受渡し又は参照を行いながら，複数の機能を逐次的に実行している。再度見直しを図り，必要に応じて更にモジュール分割を行った方がよい。

エ　これは“論理的強度”のモジュールである。関連した幾つかの機能を含み，パラメタによっていずれかの機能を選択して実行している。現状では大きな問題となっていないとしても，仕様変更に伴うパラメタの変更による影響を最小限に抑えるために，機能ごとにモジュールを分割するか，機能ごとの入口点を設ける方がよい。

解答解説

モジュールの論理的強度に関する問題である。

モジュール設計書によると、挿入処理、更新処理、削除処理の機能がコードとして、上位モジュールから渡されて、そのコードに基づいてモジュール内の処理が決まるモジュールである。複数の機能をパラメータによって制御する。

アの暗号的強度は、モジュール内の機能間に関係はなく、他のモジュールとの強い関係を持っている。

イの情報的強度は、複数の機能を持っているが、各処理に個別の入り口があるため、制御の結びつきがない。

ウの連絡的強度は、モジュール内の複数の機能を逐次的に実行する。

エの論理的強度は、モジュール内に複数の機能を持ち、パラメータにより機能を選択し実行するモジュールである。パラメータにより機能が制御される。求める答えはエとなる。

txt04031　プログラミングとテスト

txt040311 プログラミング

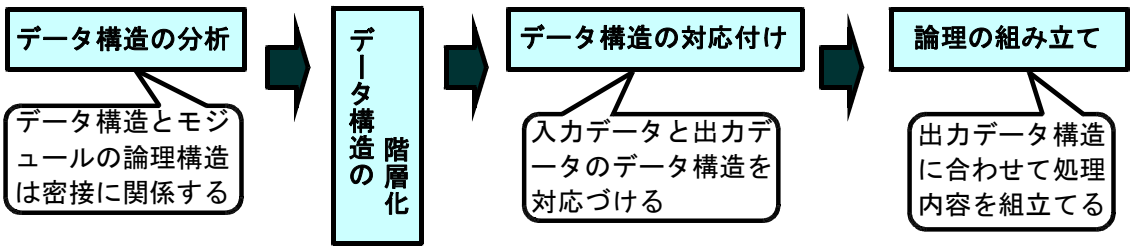
①　論理の設計

ⓐ　論理構造化の目的

㋐　モジュールの論理をやさしく組み立てることである。
㋑　複雑さを減少させ、わかりやすいプログラムを作成する。
㋒　開発の生産性と保守の拡張性の向上

ⓑ　論理設計とは

論理設計はモジュールの機能を実行する詳細な論理を作成する。わかりにくい論理はプログラミングの生産性を低くし、保守作業の負荷を大きくする。構造化定理を用いて論理を構造化する。該当モジュールが取り扱う入力データと出力データに着目して論理を組み立てる。



ⓒ　論理設計の手順

㋐　データ構造の分析

モジュールの論理構造はデータの構造と密接に関係する。入力データと出力データを明確にし、入力データと出力データを対応づける。データ構造が繰り返しの構造になっていると、モジュールの論理も繰り返しの論理構造となる。データ構造が選択の構造になっていると、論理構造も選択の論理構造になる。

㋑　データ構造の階層化

データ構造を分析し、入力データ構造と出力データ構造ごとに階層化する。データ構造の階層化を行うには、ファイルを構成するレコードを考え、各レコードのデータ項目を分解し、データ項目の意味を考えて、枠組みを行い、データ構造を作成する。

㋒　データ構造の対応づけ

入力データと出力データのデータ構造を作成し、対応関係を明確にする。関連づけはデータ構造の分析と処理機能の分析によって行う。単なるデータの移動によって完了するデータ項目や変換の論理が必要なデータ項目、加減乗除などの演算が必要な項目などの整理を行い関連づける。

㊓　論理の組立

出力データ構造にあわせて、出力条件を明確にし、処理内容を組み立てる。出力条件が満たされたときの処理内容、出力条件が満たされないときの処理内容、データの処理を行う前に必要な初期処理、データの処理が終了したときに必要な後処理などについて検討する。作成した論理は擬似コードを用いて、わかりやすく記述する。擬似コードは選択の論理、繰り返しの論理などを用いて、他の人が理解しやすいように記述する。文章の記述は簡潔になるように箇条書きする。わかりにくい部分は、補足欄を設け、補足説明を行う。

ⓓ　論理を記述する際の留意点

㋐　処理条件を明確に記述する。
㋑　論理の表現は明確に記述する。曖昧な表現は誤解を生じやすく、エラーを発生させる原因になる。
㋒　処理条件を整理し、処理条件が同じものをセグメントとしてまとめる。
㋓　データを処理、加工するセグメントと、それらを制御するセグメントに分けて、論理を設計する。
㋔　データ処理、加工セグメントは、データの移動や変換、編集、演算を中心に論理を展開する。
㋕　制御セグメントはデータ処理、加工セグメントの実行条件や実行順序の論理を組み立てる。
㋖　制御セグメントはファイルの読込や書出し、基本ファイルと取引ファイルとのマッチング処理、終了時処理の制御などがある。

②　構造化の定理

ⓐ　基本制御構造

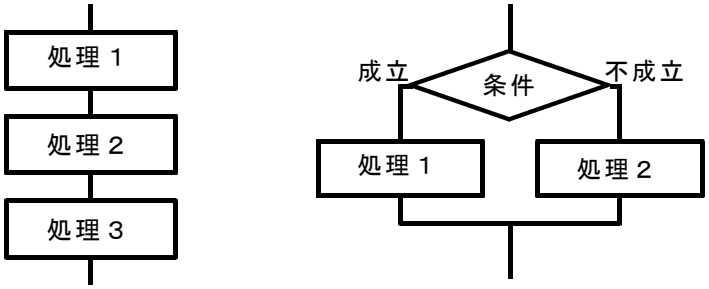
適正プログラムとは、一つの入口と一つの出口をもち、無限のループもなく、また、どんな条件でも実行されない文が存在しないものを適正プログラムという。そのためには、基本制御構造である３つの基本形の順次、選択、繰返しだけでその論理を記述する必要がある。

ⓑ　順次型

順次型は、goto文や、論理判断を含まない逐次的に実行される文だけで構成される論理構造である。処理１、処理２、処理３で実行される内容は、機能としてまとまりのある場合もあるし、詳細に記述した場合は、一つの命令を意味する場合もある。

ⓒ　選択型（if then else型）

選択型は、ある条件が成立するかどうかにより、実行する処理が異なるときに使用する論理構造である。条件が成立するかどうかを判断し、成立した場合には、処理１を実行し、不成立の場合は処理２を実行する。



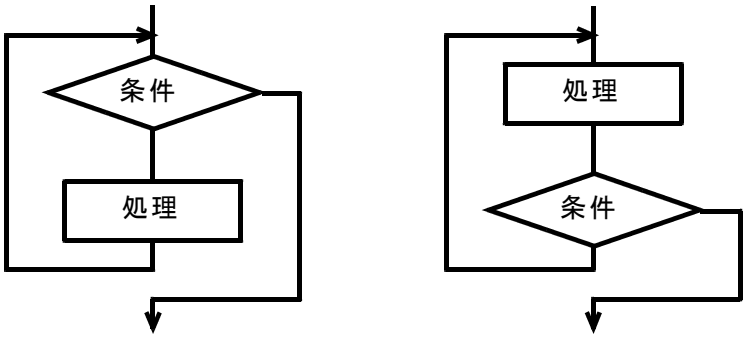
ⓓ　繰返し型

㋐　do while型

do while型は、ある条件を判断し、その条件が成立すると指定された処理を実行し、再び条件判断へ戻る論理構造である。この繰返しは条件不成立になるまで繰り返される。指定された処理が一度も実行されない場合がある。前判定型という。

㋑　do until型

do until型は、条件が成立したら次の処理に進む。条件が不成立の間、処理を繰り返す論理構造である。繰返し内の処理は、条件の内容如何にかかわらず、必ず１回は実行される。後判定型という。



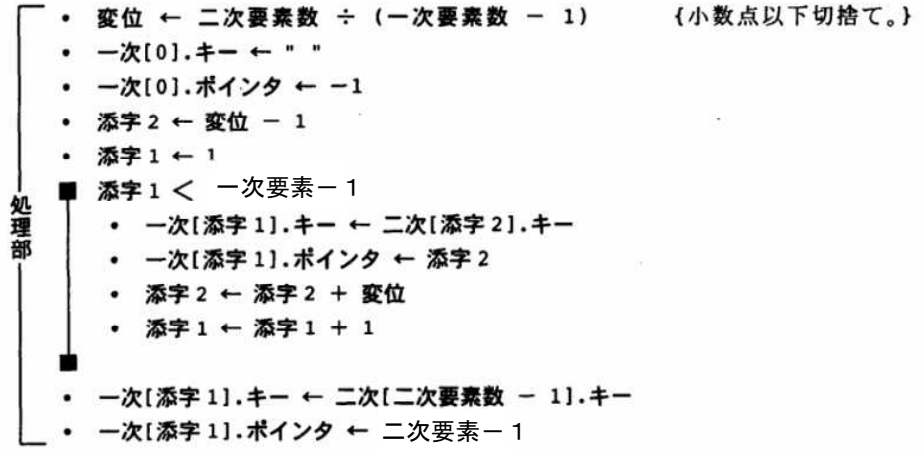
③　プログラム記述言語(擬似言語)

ⓐ　擬似言語とは

擬似言語はモジュールの論理を設計していくときの思考過程を体系化する際に使用する自然言語に近い表現法である。記述形式を表に示す。宣言部と処理部からなり、宣言部では手続き、

変数、型などを宣言し、処理部では具体的な処理内容を記述する。代入式、選択処理の条件式、繰返し処理の条件式が一定の規則に従って記述される。処理部の具体的な例を示す。

〔一次インデックス作成プログラム〕



記述形式	説　明
宣言部	手続、変数、型などを宣言する領域である。
○	手続、変数などの名前、型などを宣言する。
処理部	処理を記述する領域である。
● 変数←式	変数に式の値を代入する。
{文}	注釈を記述する。
条件式 ● 処理１ ● 処理２	選択処理を示す。 条件式が真のときは処理１を実行し、 偽のときは処理２を実行する。
条件式 ● 処理	前判定繰り返し処理を示す。 条件式が真の間、処理を実行する。

ⓑ　擬似言語の特徴

㋐　利用部門の人たちが理解できるような方法で書く。
㋑　論理展開はトップダウンで、表現は自由である。
㋒　文法上の制約を考慮する必要がない。
㋓　構造を表現するために字下げを行う。

④　判断表

ⓐ　判断表(デシジョンテーブル)とは

判断表は問題を処理するための条件と、その行動の関係を表形式で表すものである。条件と行動との対応関係を表で表現することによって、複雑な論理を一目でわかるように表現できる。

ⓑ　判断表の特徴

㋐　IF THEN ELSE型の分岐構造を整理するのに向いている。
㋑　複雑な分岐構造であっても、判断表ができていれば、その部分の論理に関しては、流れ図を書く必要がなくなる。
㋒　流れ図を作成する前に、判断表でまとめると、漏れや間違いを防ぐことができる。

ⓒ　判断表の構成要素

条件表題欄は判定すべき、すべての条件を記入する。行動表題欄は特定の条件が満足された場合に実行すべき処理のすべてを列挙する。条件記入欄は条件表題欄の記入条件に対する判定結果を記入する。行動記入欄は条件記入欄に記入した判定が満足された場合にとるべき行動の該当欄にＸ印を入れる。

条件表題欄	条件記入欄
行動表題欄	行動記入欄

ⓓ　作成上の留意点

㋐　条件表題欄と行動表題欄で記述する文章は、できるだけ簡潔で要領を得たものにする。
㋑　条件表題欄や行動表題欄では、誤解の危険性がまったくない場合以外には、略号はできるだけ使用しないこと。
㋒　二重否定による誤解を避けるために、条件表題欄と行動表題欄は肯定表現で記述する。

㊓　条件表題欄の記述内容の全体を把握しやすくし、記入漏れを防止するために、記入の順序を工夫する。

㊔　条件記入欄にはすべての可能な条件の組合せをもれなく記入する。

㊕　条件記入欄は読みやすくするために、できるだけ一貫性を保つように記入する。

㊖　ある特定の規則に該当しない無関係の条件は、条件記入欄に記入しないようにする。

⑤　ＮＳチャート

ⓐ　ＮＳチャートとは

構造化チャートともいい、ナッシュとシュナイダーマンによって開発された表記法である。論理の流れを表すのに流れ線を用いず、論理の階層化を図的に記述できる。

ⓑ　ＮＳチャートの要素

㋐　処理

処理を表す場合は長方形を用いる。長方形の中には、他の記号を入れて、入れ子構造にしてもよい。直線型の構造パターンを表現するには、順次に処理する一連の機能を一つの枠内にすべて順番に記述していく。この構造では、一つの入口と一つの出口しか許されない構造になる。

㋑　選択

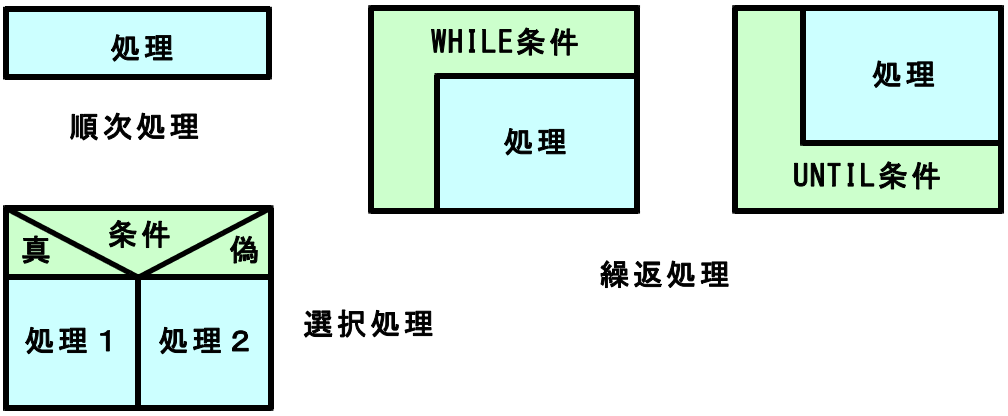
選択の論理は、上欄を三角形に区切り、上の三角形に条件を、下の二つの三角形に条件の判断結果の真・偽を入れる。条件判定欄の下にあるボックスは、真または偽の場合の処理ボックスである。IF THEN 型のように何も処理しない場合には、単に矢印を記入し通過を表したり、まったくブランクにとしておく。

㋒　繰返し

処理の繰返しのDO WHILE型では、処理ボックスの上部に条件を記述し、処理が真の間、処理が繰り返される。DO UNTIL型の場合、処理が実行された後に条件が判定される論理を表すため、処理の下に記述された条件が真の間、繰返し処理が実行される。処理ボックスはほかの処理ボックスと同じ性質であるから、入れ子構造にすることができ、入れ子になった処理も繰返し実行されることになる。

㋓　多岐選択

多岐選択は３個以上の選択処理の中から条件を満足する処理を１個選択する場合である。各条件に応じた処理は他の構造単位と同様に、処理や選択、繰返し、あるいはそれらの複合の形であってもよい。



⑥　低水準言語

ⓐ　低水準言語とは

特定のコンピュータの構造や動作の仕組みに強く依存している言語で、個々の命令は、各種レジスタ、ポインタ、カウント、フラグ類などを直接操作する機能をもっている。プログラム作成者は使用するコンピュータのハードウェアを熟知している必要がある。一つの命令が機械語と１対１に対応するプログラム言語で、アセンブラ言語が相当する。

ⓑ　低水準言語の特徴

㋐　プログラムの実行速度は速い。
㋑　プログラムの作成にハードウェア知識が必要である。
㋒　ハードウェアの変更等による影響を受けやすい。
㋓　プログラムの修正などは、会話的に行うことができない。
㋔　プログラム開発の生産性は高水準言語に比べて低い。

⑦　高水準言語

ⓐ　高水準言語とは

コンピュータの利用者がハードウェアの特性を意識することなくプログラムの作成が可能なプログラム言語で、作成したプログラムは、文書性が高く、比較的容易に解読できる。次の二つの言語に大別できる。

㋐　手続き型言語
㋑　非手続き型言語

ⓑ　高水準言語の特徴

㋐　処理の手順を記述しやすく、問題解決に適している。
㋑　低水準言語に比べて、自然語に近い構造をしている。
㋒　特定のハードウェアに左右されない。
㋓　複数の機械語の命令を１つの命令で記述できるため、プログラムのステップ数が少ない。

ⓒ　手続型言語

目的とする仕事の手順を記述する言語である。仕事の手順やアルゴリズムの作成に使用され、命令を書いた順に実行する。ＣＯＢＯＬ、Ｃ言語、ＢＡＳＩＣなどが該当する。

ⓓ　手続型言語の特徴

㋐　プログラマの考える仕事の処理手順やアルゴリズムを自由かつ詳細に記述できる。
㋑　命令記述の仕方は、基本的にアルゴリズムに対応している。
㋒　開発の生産性は、非手続き型に比べると低い。
㋓　プログラム作成にあたっては、言語の知識と仕事の詳細な処理手順で表現したり、アルゴリズムで表現する技術が必要である。

ⓔ　非手続型言語

非手続型言語は、仕事の処理手順やアルゴリズムを意識することなく、仕事の機能を定形化する技術が主体となっている言語である。仕事の処理事項を指示したり、簡単な命令やパラメタを与えることでプログラムを作成する。関数型言語、論理型言語、オブジェクト指向型言語、表形式型言語などの非手続き型言語は機能中心に記述する仕様になっている。

ⓕ　非手続型言語の特徴

㋐　アルゴリズム表現などの知識や技術が必要なプログラミングを簡略化できる。
㋑　処理手順が定形化されているため、ソフトウェア開発の生産性が高い。
㋒　処理手順が定形化され、規定の関数やアルゴリズムを使用するため、プログラムの自由度が低い。
㋓　非手続き型言語は、言語の性質上、手続き型言語よりさらに高水準の言語である。

ⓖ　オブジェクト指向言語

オブジェクト指向言語は、プログラムが扱う対象をものであるオブジェクトとしてとらえて、プログラムを作成する。外界の対象領域は、実体であるオブジェクトで構成される。手続きや関数を記述するのではなく、オブジェクトの状態や振舞いを記述する方式である。

各オブジェクトは外部から与えられた作用によって、その内部状態を遷移する特性を持って

いる。カプセル化機能、メッセージパッシング機能、インヘリタンス機能などを有する。Small talk、C++、Javaなどがある。

ⓗ　オブジェクト指向言語の特徴

㋐　インスタンス機能はクラスから効率的にオブジェクトを生成する。同一のクラスに属するオブジェクトは、共通のメソッドを持つ。

㋑　カプセル化機能はデータとメソッドを一体化した機能で、オブジェクト指向プログラミングの生産性と信頼性を向上させる。オブジェクトの独立性が高く、開発時や保守時の波及効果が及ばず、効率のよい作業工程が実現する。

㋒　メッセージパッシング機能はオブジェクト同士が互いに情報をやり取りし、協調的に問題解決を図るために、メッセージを送受信する機能、実行制御の順序やデータ構造について考量する必要なく、協調的に動作する。

㋓　インヘリタンス機能は親クラスで指定された性質は、子クラスに引き継がれる。子クラスでは、新たなメソッドを追加することができる。これを差分プログラミングという。対象となる適用業務処理毎に、既存のクラス階層をライブラリとして作成し、再利用が可能になる。

ⓘ　第四世代言語

プログラム開発の知識・経験のない人でも、簡単にプログラムを作成できるようなプログラム言語で、業務の処理内容をパラメータによって指示する。問題解決のための処理手順を自動生成し、少ない命令でプログラムを作成する。

ソフトウェア開発部門は、新規開発、必要不可欠な仕様変更、開発後のトラブル対策等の保守作業の増大は、バックログを増大させる傾向にあり、この対策のために第四世代言語が必要となった。

ⓙ　第四世代言語の特徴

㋐　エンドユーザの立場に立って開発された言語である。

㋑　言語に精通したプログラマでなくても、容易に使用できる。

㋒　言語によっては、プログラムのステップ数が従来の言語に比べて少なくなる。

㋓　デバッグが比較的容易である。

㋔　開発・保守の期間の短縮を図ることができる。

㋕　短期間で言語の使用方法を修得できる。

㋖　作成されたソフトウェアを容易に理解できる。

㋗　構造化プログラミング技法の原則に沿っている。

㋘　オンライン操作が可能であり、データベース管理システムの使用ができる。

ⓚ　エンドユーザ言語とは

エンドユーザや一般利用者がプログラムを作成することなく、定型業務処理や表、グラフの作成を行うことができる目的に開発された言語である。エンドユーザ言語はソフトウェアの生産性と要求された情報を即座に提供できる特徴がある。

イベント駆動型で、利用者がキーボードやマウスで入力すると、直ちに処理を開始する。ＧＵＩを利用した高機能な開発環境が整備されており、アプリケーションプログラムのカスタマイズ用としても利用される。

①　エンドユーザ言語の特徴

㋐　特別なコンピュータの知識やプログラミング技術を必要としない。
㋑　言語の修得がほかの高水準言語に比べて容易である。
㋒　言語のもつ規則性を知れば、エンドユーザや素人でも自由に使用できる。
㋓　短時間でプログラムの作成ができるため、開発の生産性が高い。
㋔　定形化された機能以外の要求には対応できないことが多い。
㋕　既存のモデルやアルゴリズムの変更には、困難を伴うことが多い。
㋖　容易に開発できる反面、大量のデータ処理や高性能の処理、複雑な処理などには向かないことが多い。

ⓜ　特殊問題向き言語とは

問題解決に長時間を要したり、複雑な高度な計算を必要とする特定分野に限った業務処理を目的として開発された言語である。離散型シミュレーション用プログラムの作成に広く利用されるＧＰＳＳ、流体力学や天文学などの複雑・高度な数式処理を行うためのプログラム言語であるFORMAC、建築物の構造解析・設計に用いられるプログラム言語のＣＯＧＯがある。

ⓝ　特殊問題向き言語の特徴

㋐　適用分野が限定されているため、その範囲内では汎用のプログラム言語に比べて、より速く、確実にプログラムを作成することができる。
㋑　メーカがソフトウェアパッケージとして提供しているため、利用者にとって、信頼性・開発コストなどの面で有利である。
㋒　利用する分野の専門知識が必要である。
㋓　適用分野以外のプログラムには使用できない。

⑧　各種プログラム言語

言語名	特　　徴
Fortran	●科学技術計算向きの複雑な計算を高速に処理するのが目的のコンパイラ言語で、1957年ＩＢＭ社で開発された。 ●主プログラムは、実行文、非実行文、END文、副プログラムはサブルーチン文、関数文からなり、データ型は、整数型、実数型、複素数型、論理型、文字型で、算術式、論理式などが使用できる。 ●言語の構造が簡単で、数値的な問題や論理演算などのアルゴリズムの表現が可能で、三角関数、対数関数など関数が豊富である。
COBOL	●事務処理用に使用するコンパイラ言語で、1959年CODASYL委員会で開発された。 ●ファイル構造の定義、効率的なファイルの入出力、整列機能、報告書作成機能、COPY機能が特徴である。プログラムは、見出し部、環境部、データ部、手続き部からなる。 ●プログラムの文書性が高く、作成保守が容易である。英語に近い表現で、わかりやすく、記述しやすい。
Ｃ	●UNIXのＯＳを開発するために設計されたコンパイラ言語。 ●豊富な演算子とデータ型、制御構造を持ち、ポインタを使用してアドレスの概念が実現できる。低水準言語に近いビット操作も可能である。 ●文の中に式を記述することも可能で、プログラムの記述はフリーフォーマットで、構造化機能もある。 ●ミニコンピュータ、ワークステーション、パーソナルコンピュータなど様々なコンピュータで広く利用されている。 ●Ｃ言語にオブジェクト指向の機能を追加したC++も普及している。 ●移植性の高さとモジュール開発の容易さ ●パッケージプログラムの開発に利用する。 ●開発済みモジュールの再利用を促進 ●開発効率の向上 ●プログラム品質の改善
BASIC	●パーソナルコンピュータの高水準言語として普及している。 ●会話型でプログラムが作成でき、命令の記述の修得が容易である。個々の命令が短く、文の先頭に文番号を持つ。互換性、移植性に劣る。 ●プログラムの翻訳と実行が同時に行われるインタプリタ方式で、コンパイラ言語に比べて実効速度が遅い。
ＲＰＧ	●報告書を作成することを主目的とした言語。 ●各種のデータ処理項目をパラメータの指定で行う。アルゴリズムを意識することなく、プログラム作成の知識がなくてもプログラムが作成できる。プログラム作成の生産性が高い。

言語名	特　　徴
ALGOL	●科学技術計算用で、ヨーロッパを中心に使用された。 ●ＢＮＦで言語が定義されており、ブロックと複合文で構造的に記述でき、データや変数、手続きは宣言で定義する。再帰呼び出しが可能である。
ＰＬ／１	●科学技術計算、事務処理の両方に使用できる言語 ●プログラムはブロック構造を持ち、データの型が豊富で、リスト処理、テキスト処理など様々な問題処理に利用される。
Pascal	●科学技術計算用で、構造化プログラミング指向の言語である。 ●フリーフォーマット形式でプログラムの記述ができ、データ型は、配列やレコード、ポインタなどを組み合わせた型が自由に構成できる。手続きや関数で再帰呼び出しができる。
ＬＩＳＰ	●人工知能関係のソフトウェア開発、数式処理のためのプログラム言語。 ●数式などをリストというデータ構造で表現する。関数型言語であり、標準関数とユーザ定義関数を組み合わせて、帰納的、再帰的定義によって記述できる。
Prolog	●人工知能関係のソフトウェア開発用のプログラム言語で、推論機能をもつ論理型言語である。 ●プログラムは事実と推論規則から構成されている。問題を解析し、論理的な推論により解を導く方法で行う。
ＡＰＬ	●数学的、論理的な関係の定義ができる言語で、関数型言語である。 ●特異な文字を使用して、プログラムの記述を行うため、ＡＰＬ記号を備えたキーボードが必要である。
Ada	●Pascalを言語の模範としたシステム記述言語である。 ●パッケージというプログラム単位で大型のソフトウェアを作成する。
ＳＱＬ	●関係データベースを構築し、アクセスするための言語である。 ●コマンド形式で入力することで、データの検索、更新、表の定義などを行うことができる。
Perl	●テキスト処理用のインタプリタ言語で、WWWサーバ上で動くＣＧＩプログラムの標準言語である。
ＳＧＭＬ	●文書の論理構造や意味構造をタグマークで記述する言語である。 ●電子文書をコンピュータが理解する記述が可能であるが、複雑である。
ＨＴＭＬ	●WWWで使用するハイパーテキストを記述する言語である。 ●ＳＧＭＬを拡張して作られた言語で、タグを利用し、画像ファイルやリンク先などを簡単に使用できる。
ＸＭＬ	●ＨＴＭＬのリンク機能を拡張して、ＳＧＭＬをインターネット向けに最適化した言語である。 ●ＨＴＭＬに比べて、利用者の自由度が高く、柔軟性があり、電子商取引のデータ交換、企業の書類保管などに応用される。

言語名	特　　徴
Smalltalk	●オブジェクト指向言語で、オブジェクトやメッセージ表現する言語。 ●オブジェクト、メソッド、クラスなどを使用して、対話形式で、簡単な言語構文でプログラムの記述ができる。
C++	●Ｃ言語にオブジェクト指向の概念を付加した互換性のある言語 ●クラス、継承などの概念がある。
Visual Basic	●ビジュアル言語で、BASICをWindows環境で動作するようにした言語。 ●Windows環境でアプリケーションプログラムを開発する。
Java	●インタプリタ形式のオブジェクト指向言語で、仮想マシン上で動作する。 ●インタネット上のWWWブラウザで使用し、インタラクティブな操作が可能となる。JavaＶＭが搭載されておれば、プラットフォームに依存しない特徴があり、セキュリティとネットワーク機能が強化されている。 ●ガベージコレクション機能を備えている。 ●ポインタの概念がない。 ●ネットワーク上のプログラムをダウンロードし、Ｗｅｂ上で動かすことができる。このプログラムをJavaアプレットと呼ぶ。 ●実行速度は遅い。 ●インターネットや分散システム環境で利用されている。 ●単一継承のみで、多重継承はサポートされていない。

例題演習

プログラムの制御構造に関する記述のうち，適切なものはどれか。

ア　“後判定繰返し”は，繰返し処理の先頭で終了条件の判定を行う。
イ　“双岐選択”は，前の処理に戻るか，次の処理に進むかを選択する。
ウ　“多岐選択”は，二つ以上の処理を並列に行う。
エ　“前判定繰返し”は，繰返し処理を１回も行わないことがある。

解答解説

構造化定理に関する問題である。

構造化定理は、順次型、選択型、繰り返し型の基本制御構造を用いる。繰り返し型には前判定と後判定がある。

アの後判定繰返しは、繰り返し処理の後で終了条件の判定を行う。従って、処理は必ず１回は実行される。先頭で終了条件の判定を行うは誤りである。

イの双岐選択は、二つの処理のうちどちらを実行するかを選択することである。前に戻るか、次の処理に進むかの判定ではない。

ウの多岐選択は、二つ以上の処理のうちどれか一つの処理を実行する場合の選択である。並列に処理をするのではない。

エの前判定繰り返しは、最初の繰り返し判定で偽になると繰り返しに入らないことになる。繰り返し処理を１回も行わないことになる。求める答えはエとなる。

例題演習

プログラムの工程を説明しているのは、次のうちのどれか。

ア　システム化計画、プロジェクト実行計画、要求定義の３つに分けられる。
イ　プログラムの構造化設計を行う。プログラムの分割技法を使用して、プログラムを複数のモジュールに分割する。
ウ　プログラムを意識しながら、システムの機能を分割し、構造化を行う。
エ　モジュール設計、単体テスト計画、コーディング、単体テストがある。

解答解説

プログラミング工程の作業内容に関する問題である。

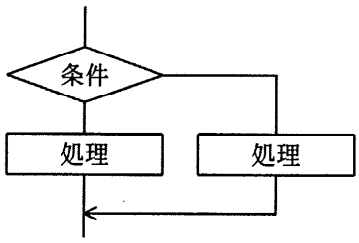
プログラミングでは、プログラム設計段階のモジュール分割の後を受けて、モジュール内の論理設計、コーディング、モジュールテスト計画、モジュールテストが行われる。

アは基本計画、イはプログラム設計、ウは内部設計、エはプログラミングで、求める答えはエとなる。

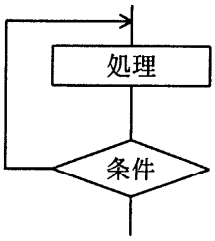
例題演習

プログラムの制御構造のうち，While型の繰返し構造はどれか。

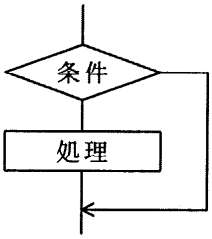
ア



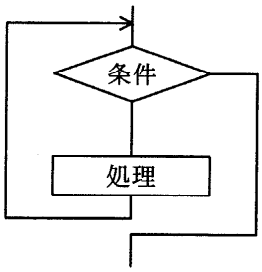
イ



ウ



エ



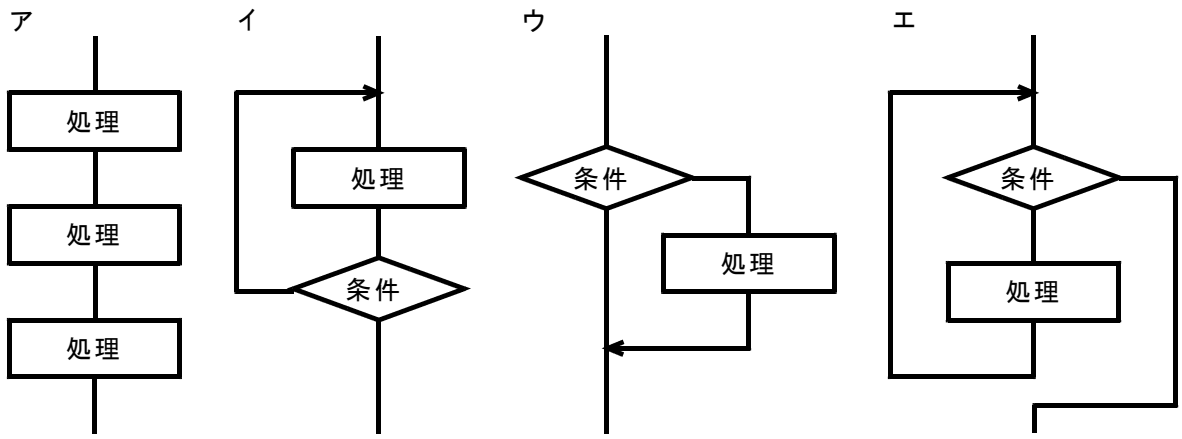
解答解説

構造化定理に関する問題である。

ア、ウは選択型、イは繰返し型のdo while型、エは繰返し型のwhile 型である。求める答えはエとなる。

例題演習

次のプログラムの制御構造のうち、選択構造はどれか。



解答解説

プログラムの制御構造に関する問題である。

基本となるプログラムの制御構造に、順次構造、選択構造、繰返し構造がある。順次構造は制御構造を持たない処理が単独または連続して現れる構造である。選択構造は制御条件式の判定結果によって、真か偽のいずれか一方が選択される制御構造である。繰返し構造は同じ処理を繰返し実行する制御構造で、前判定型と後判定型がある。

アは順次構造、イは繰返し構造の後判定型、ウは選択構造、エは繰返し構造の前判定型である。求める答えはウとなる。

例題演習

プログラム図式の一つである構造化チャートを説明する記述として，最も適切なものはどれか。

ア　ＧＯＴＯを表現する方法をもたず，論理の階層化を図的に表現できる。モジュール内の論理を構造化して表現するのに都合の良い図式である。

イ　制御システムなどの内部設計に広く用いられる手法である。データの流れを示せないので，ほかの図式と併用するのが合理的である。

ウ　データの流れを表すのが容易な図式である。処理の手順を表しにくい。

エ　モジュール構造図を示す図式目次と，各モジュールの機能を入力・処理・出力の形式で記述した図からなる。構造化手法の特徴であるトップダウン設計向きの図式である。

解答解説

構造化チャートに関する問題である。

構造化チャートは、機能間の従属関係を階層的に表し、パラメータの記述ができるため機能間のインタフェースが理解しやすく、ループ、判定などの手続きを表現することができるため、プログラムの論理構造が理解しやすいなどの特徴がある。

アは構造化チャート、イは状態遷移図、ウはＤＦＤ、エはＨＩＰＯで、求める答えはアとな

る。

例題演習

順次、選択、反復の制御構造を表すボックス図によってプログラム構造を記述する設計手法はどれか。

ア　スキーマ　　イ　ＮＳチャート　　ウ　ＤＦＤ　　エ　疑似コード

解答解説

プログラム構造を記述するＮＳチャートに関する問題である。

アのスキーマは、データベースの仕様を記述したもので、概念スキーマ、外部スキーマ、内部スキーマの３種類がある。

イのＮＳチャートは、構造化プログラミングに使用される図式の１つで、全体を四角で表し、順次、選択、反復を表すボックス図の記号を組み合わせて論理を記述し、構造化を図る方式である。求める答えはイである。

ウのデータフローダイヤグラムは、データの流れを図式化したもので、データフロー、プロセス、データストア、外部の４種類の基本要素で構成される。

エの疑似コードは、アルゴリズムを記述するために用いる人の言葉に近い言語である。

例題演習

プログラム言語の特徴に関する記述のうち，適切なものはどれか。

ア　ＣＯＢＯＬは，事務処理に適しており，インタプリタ方式で実行される。、
イ　Ｃは，システム記述に適しており，そのプログラムの実行にはコンパイルが必要となる。
ウ　Javaは，言語仕様がプラットフォームに依存しており，インタプリタ方式で実行される。
エ　Perlは，クライアント上で動作するプログラムの記述に適しており，そのプログラムの実行にはコンパイルが必要となる。

解答解説

プログラム言語の特徴に関する問題である。

Ｃ言語は、ＯＳなど制御系プログラムなどの開発に適したプログラム言語である。メモリを直接扱うなどアセンブラと同等レベルのハードウェアに近い制御を記述することができる。

アのＣＯＢＯＬは、事務処理向けのプログラム言語で、コンパイラ言語である。インタプリタは誤りである。

イのＣに関する記述は適切である。求める答えはイとなる。

ウのJavaは、オブジェクト指向のプログラム言語で、ＯＳやパソコンの機能に依存することなく動作する。Javaのソースコードをコンパイルすると、バイトコードの中間コードに変換され、このコードをJava仮想マシンが解釈して、プログラムを実行する。Ｗｅｂブラウザ上で動作するプログラムをJavaアプレット、Ｗｅｂブラウザに関係なく動作するプログラムをJavaアプリケーションと呼ぶ。言語仕様がプラットフォームに依存するは誤りである。

エのPerlは、テキストやファイル処理に適したインタプリタ型のプログラム言語で、WWWサーバでは、ＣＧＩプログラムの標準言語として活用されている。クライアントではなくサーバ上で動作し、コンパイラではなくインタプリタである。

例題演習

Java言語で作成されたプログラムは，異なるハードウエアや異なる基本ソフトウェア上で実行可能なことが特徴である。この実現に関連するものとして適切なものはどれか。

ア　ＪＩＴ(Just in Time)コンパイラ　　イ　仮想マシン
ウ　クロスコンパイラ　　エ　リバースエンジニアリング

解答解説

Javaの仮想マシンに関する問題である。

アのＪＩＴコンパイラは、Javaプログラムのバイトコードを、実行環境の機種に最適化されたネイティブ・コードに変換するコンパイラである。

イの仮想マシンは、JavaＶＭで、Java言語で作成したソースプログラムは、Javaコンパイラにより、バイトコードと呼ぶ中間コードに変換されて配布され、JavaＶＭで実行される。JavaＶＭが組み込まれている環境であれば、コンピュータの機種やＯＳに関係なく動作可能である。求める答えはイとなる。

ウのクロスコンパイラは、他のコンピュータシステムで動作する機械語を生成するためのコンパイラである。開発環境のない組込用コンピュータのソフトウェア開発に使用される。

エのリバースエンジニアリングは、互換製品を作る目的で、オリジナル製品を解析して設計情報などを調査する手法である。

例題演習

プログラム言語に関する記述のうち、Java言語の説明として適切なものはどれか。

ア　1970年代に開発されたインタプリタ型のオブジェクト指向言語であり、エディタやデバッガなどの統合開発環境やＯＳの機能などを含む。
イ　Ｃ言語にクラスやインヘリタンスといったオブジェクト指向の概念を取り入れたものであり、Ｃ言語の上位互換性をもつ。
ウ　Ｗｅｂで用いられるマーク付き言語であり、タグによって文書の構造を識別する。テキストや動画などを関連づけたハイパテキストを作成する。
エ　ブラウザと連動して動作するアプレットなどを作成できる。このアプレットは、仮想マシンを実装する環境上であれば、どこでも実行できる。

解答解説

Java言語に関する問題である。

Java言語は、オブジェクト指向のプログラム言語で、パソコンやプリンタ、携帯電話、テレビなどの家電製品など様々な分野で利用が考えられている。Javaで記述したプログラムをコン

パイルするとJavaＶＭ用のバイトコードが生成される。JavaＶＭを搭載しておれば、どのプラットフォームでも動かすことができる。

アはsmalltalk、イはＣ＋＋、ウはＨＴＭＬ、エはJavaである。求める答えはエとなる。

例題演習

プログラム言語Javaに関する記述として，適切なものはどれか。

ア　インターネットや分散システム環境で利用されている，オブジェクト指向のプログラム言語である。

イ　テキスト(文字)で記述が可能な，自然言語に近いプログラム言語である。

ウ　テキストやファイルの処理に適しており，連想配列とパッケージを結び付けることが可能なプログラム言語である。

エ　ハイパテキストを記述する言語であり，アンカーというタグを用いて別の文書とリンクが可能なプログラム言語である。

解答解説

プログラム言語に関する問題である。

アはJava言語、イは疑似言語、ウはPerl、エはＨＴＭＬである。求める答えはアとなる。

例題演習

JavaＶＭが稼働している環境だけがあれば，Ｗｅｂブラウザや Ｗｅｂサーバがなくても動作するプログラムはどれか。

ア　JavaScript　　イ　Javaアプリケーション

ウ　Javaアプレット　　エ　Javaサーブレット

解答解説

Javaアプリケーションに関する問題である。

アのJavaScriptは、オブジェクト指向のインターネット用スクリプト言語で、ユーザの操作に合わせてウェブページの表示内容を変えたり、アンケートなどでフォームに入力したデータをチェックするなどＨＴＭＬで実現できない機能をウェブページ上で実現する。

イのJavaアプリケーションは、Javaで作成されたソフトのうち、ユーザのコンピュータに組み込んで、ウェブブラウザと独立して単体で動作させることが可能なアプリケーションである。ＶＭが稼働しておれば動作する。動作しているコンピュータのファイルの読み込み、書き換えは可能である。求める答えはイとなる。

エのJavaアプレットは、Javaで作成されたソフトで、ＨＴＭＬに組み込まれてウェブサーバからダウンロードし、ウェブブラウザ上で実行されるものである。ユーザのコンピュータのファイルの読み書きや他のアプリケーションを起動できない。

ウのJavaサーブレットは、サーバ上で動作するJavaアプレットで、Ｗｅｂサーバ上でデータベースの検索などクライアントに対してさまざまなサービスを提供できる。ＣＧＩに比べてＷ

ｅｂサーバのパフォーマンスが落ちないメリットがある。

例題演習

Javaのプログラムにおいて，よく使われる機能などを部品化し，再利用できるようにコンポーネント化するための仕様はどれか。

ア　JavaBeans　　イ　JavaScript
ウ　Javaアプリケーション　　エ　Javaアプレット

解答解説

JavaBeansに関する問題である。

アのJavaBeansは、Javaのプログラムで構成されたソフトウェアをアプリケーション部品として取り扱うための規約、仕様である。求める答はアである。

イのJavaScriptは、オブジェクト指向のインターネット用スクリプト言語で、ユーザの操作に合わせてウェブページの表示内容を変えたり、アンケートなどでフォームに入力したデータをチェックするなどＨＴＭＬで実現できない機能をウェブページ上で実現する。

ウのJavaアプリケーションは、Javaで作成されたソフトのうち、ユーザのコンピュータに組み込んで、ウェブブラウザと独立して単体で動作させることが可能なアプリケーションである。

エのJavaアプレットは、Javaで作成されたソフトで、ＨＴＭＬに組み込まれてウェブサーバからダウンロードし、ウェブブラウザ上で実行されるものである。

例題演習

ＸＭＬに関する記述のうち，適切なものはどれか。

ア　ＨＴＭＬを基にしてその機能を拡張したものである。
イ　ＸＭＬ文書を入力するためには専用のエディタが必要である。
ウ　文書の論理構造と表示スタイルを統合したものである。
エ　利用者独自のタグを使って文書の属性情報や論理構造を定義することができる。

解答解説

ＸＭＬに関する問題である。

ＸＭＬは、WWWで使われる文書の構造を定義するための言語であり、ＨＴＭＬと異なり、文書構造の指定に使うタグを独自に定義できる特徴がある。タグを手がかりにブラウザ側で加工がしやすくなるため電子商取引などで使用されている。

アのＨＴＭＬを基にして機能拡張したのではなく、ＳＧＭＬを基にしている。

イの文書の入力は、文書を通常使用するテキストで表現するため、テキスト文字の入力可能な汎用のエディタやテキスト形式の入力用ソフトウェアで可能である。

ウの文書の論理構造と表示スタイルは分離することが可能で、１つの文書から表現媒体の種類によって異なる表現が可能となる。

エの利用者独自のタグを使って文書の属性情報や論理構造を定義することができる記述は適

切な内容である。求める答えはエとなる。

例題演習

ＸＭＬの特徴のうち，最も適切なものはどれか。

ア　ＸＭＬでは，ＨＴＭＬにWebページの表示性能の向上を主な目的とした機能を追加している。
イ　ＸＭＬでは，ネットワークを介した情報システム間のデータ交換を容易にするために，任意のタグを定義することができる。
ウ　ＸＭＬで用いることができるスタイル言語は，ＨＴＭＬと同じものである。
エ　ＸＭＬは，ＳＧＭＬを基に開発されたＨＴＭＬとは異なり，独自の仕様として開発された。

解答解説

ＸＭＬの特徴に関する問題である。

アのＨＴＭＬとの対応は、表示性能の向上ではなく、表現の自由度の向上である。

イの情報システム間のデータの交換は任意のデータをＨＴＭＬと同様に容易に送受信できる。正しい記述である。求める答えはイとなる。

ウのスタイル表現は、ＳＧＭＬの文法を簡素化した文書であり、ＳＧＭＬよりは習得しやすく、ＨＴＭＬよりは柔軟性があり、ＳＧＭＬとＨＴＭＬの中間に位置する文書構造である。。

エの開発の基はＨＴＭＬと同じＳＧＭＬである。

例題演習

ＸＭＬ文書を構成する最小単位である要素の定義方法に関する記述のうち，適切なものはどれか。

ア　開始タグと終了タグが対になって構成され，どちらのタグも省略できない。
イ　データを開始タグと終了タグで囲んで構成するが，データがないこともある。
ウ　一つのＸＭＬ文書には，階層構造を表すために複数のルート要素を定義できる。
エ　要素の種別を表すために注釈情報を付加して，これを要素名として識別する。

解答解説

ＸＭＬに関する問題である。

アの開始タグ、終了タグの省略は、ＨＴＭＬの場合には、終了タグの省略は可能であるが、ＸＭＬでは省略は認められない。ただし、空要素タグの場合は、＜要素名／＞で表現する。

イのデータは開始タグと終了タグで囲んで構成するが、データがないこともあるは適切な記述である。求める答えはイとなる。

ウの階層構造はルート要素と呼ばれる最上位層の要素は必ず１つでなければならない。

エのコメントは自由に使用することができるが、タグ中にコメントを記述することは許されていないため、要素名として識別に使用することはできない。

txt040312 プログラムの実行

① 言語プロセッサの役割

ⓐ 言語プロセッサとは

自然言語に近いプログラム言語で記述されたプログラムを、コンピュータが直接理解できる機械語に変換する。プログラム言語で書かれたソースプログラムを入力して、特定の機種のプロセッサにあったオブジェクトプログラムを出力する役割を持っている。言語プロセッサによっては、機械語に翻訳せずに、中間言語に翻訳することがある。中間言語は原始プログラムと機械語との間の中間過程の言語で、原始プログラム言語→中間言語→機械語のプロセスを利用して翻訳される。

ⓑ 言語プロセッサの特徴

㋐ プログラムを人工言語でコーディングできる。
㋑ コンピュータを使用して、能率的に機械語に翻訳できる。
㋒ 用途に応じて、プログラム言語を使い分けることができる。

ⓒ 原始プログラムと目的プログラム

原始プログラム(ソースプログラム)は、ＣＯＢＯＬやＣなどのプログラム言語で作成された、機械語になる元のプログラムで、機械語に変換する前の人間が理解できるプログラムである。原始プログラムをコンパイラを使用して機械語のプログラムに翻訳したプログラムをオブジェクトプログラム(目的プログラム)という。

② 各種言語プロセッサ

ⓐ 言語プロセッサの分類

言語プロセッサは、次の２つに分類できる。

㋐ トランスレータ
❶ アセンブラ
❷ コンパイラ
❸ ジェネレータ

㋑　インタプリタ

ⓑ　アセンブラ、コンパイラ、インタプリタ、ジェネレータ

㋐　基本機能と特徴

名称	機能と特徴
アセンブラ	●アセンブラ言語で記述されたプログラムを機械語に翻訳するプログラムである。マクロ命令の展開と機械語命令の生成の機能に分けられる。 ●特徴は翻訳結果をプログラムライブラリに格納するため、繰り返し使用が可能である。アセンブラ言語のプログラムだけが翻訳対象になる。プログラムの修正は会話的に行うことができない。
コンパイラ	●手続き型言語で記述された原始プログラムを機械語または中間言語に翻訳するプログラムである。字句解析、構文解析、意味解析、コード生成、目的プログラムの最適化の５機能で構成される。 ●特徴は対話処理による翻訳はできない。目的プログラムはプログラムライブラリに格納されるため、繰り返し使用が可能である。異なるプログラム言語で記述されたプログラムを結合できる。完成した原始プログラム全体を翻訳してはじめて実行できる。
インタプリタ	●高水準言語で記述された原始プログラムを解釈し実行する言語プロセッサで、プログラムの命令を１つずつ解釈して実行する方式である。対話型言語によるプログラム作成に向いている。機能は命令の取り出し、機械語または中間語に置換、当該命令の実行からなる。 ●特徴は、コンパイラのものに比較して処理速度が遅い。完成している部分だけを実行できる。プログラム開発を効率的に行える。小さなプログラム開発に向いている。プログラムの修正には柔軟性がある。
ジェネレータ	●非手続き型言語で記述されたプログラムから機械語プログラムを作り出す機能を持っている。ＲＰＧが代表的なものである。 ●特徴は使用目的に合わせて、入力・処理・出力に必要な要件をパラメータで与える。アルゴリズム　は記述しない。アルゴリズムがブラックボックスのため非定型的な仕様には適さない。定型の事務処理には開発の生産性を上げることができる。

㋑　コンパイラーとインタプリタの比較

コンパイラの機能は字句解析→構文解析→意味解析→コード生成→目的プログラムの最適化の順に次のように進める。

❶　プログラムを構成する単語を分析する。(字句解析)

❷ プログラムの文法的な誤りをチェックし、さらに解析を進める。（構文解析）
❸ プログラムの意味を解析する。（意味解析）
❹ 機械語のコードをつくる。（コード生成）
❺ 処理の最適化を図る。（最適化）

インタプリタは高水準言語で書かれた実行プログラムを、１行ずつ解釈しながら実行していくためのプログラムで、次のような作業を行う。
❶ プログラムを１行ごとに解釈し、中間コードに変換する。
❷ 中間コードに対応する機械語の命令を呼び出す。
❸ ＣＰＵが呼び出した機械語の命令を実行する。

ⓒ　プリプロセッサ

プリプロセッサは、プログラムの作成能力を上げるために、コンパイラで翻訳できない拡張した疑似命令を使用して作られた原始プログラムを、前もって解読可能な命令に置き換えるプログラムである。Ｃ言語などのコンパイラで用いられ、コンパイル前にマクロの展開やヘッダファイルの挿入などの処理を行う。

ⓓ　クロスコンパイラ

他のコンピュータシステムで動作する機械語を生成するためのコンパイラである。動作速度が遅いコンピュータや完成していないコンピュータ、開発環境のないコンピュータなどのソフトウェアを開発する際に、汎用のコンピュータでターゲットマシン用のプログラムのコンパイルを行い、機械語を生成する。

ⓔ　コンパイラ・コンパイラ

プログラミング言語の仕様を与えると、その言語のコンパイラを生成するプログラムである。

ⓕ　トランスレータ

ある機種用に開発された原始プログラムを他の機種用に変換するためのプログラムである。あるプログラム言語で作成された原始プログラムを、別のプログラム言語の原始プログラムに変換する場合に使用することもある。

ⓖ　シミュレータとエミュレータ

㋐　シミュレータ

シミュレータはある計算機で実行可能な目的プログラムの命令を１つずつ解釈し、他の計算機の命令に変換するプログラムである。異なるコンピュータ向けに書かれたプログラムを実行できるように作られた、模擬的な実行環境を生成するソフトウェアである。

㋑　エミュレータ

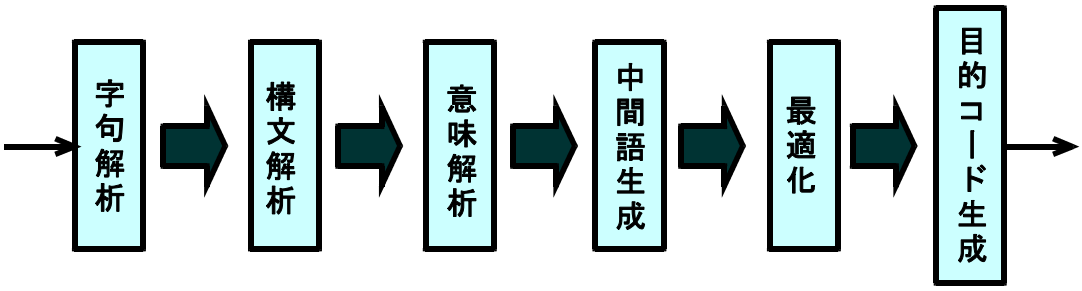
利用しているコンピュータシステムが、他のシステムと同じデータを受入、同じコンピュータプログラムを実行し、同じ結果が得られるように、他のシステムと等価な動作をするハードウェアもしくはソフトウェアである。ゲームソフトの開発を環境が整備されたパーソコンで行い、ゲームマシンと等価に実行させる場合などに利用する。エミュレータはシミュレータの機能のプログラムをファームウェアで実現したものである。

③　コンパイル技法

ⓐ　コンパイル

高水準言語で記述された原始プログラムを、コンピュータで実行することを目的とした目的プログラムに変換することをコンパイルという。変換前のプログラムを原始プログラム、変換後のプログラムを目的プログラムと呼ぶ。

ⓑ　コンパイラ



㋐　コンパイラとは

コンパイラは、ある種のプログラム言語で記述されたソースプログラムモジュールの文字列をコンピュータ内部に読み込んで、２進化符号列のオブジェクトモジュールを出力する自動変換機能を提供する。

㋑　コンパイルのフェーズ

字句解析、構文解析、意味解析、最適化、コード生成などのいくつかの処理単位別のフェーズから構成される。原始プログラムを字句解析で単語に分解し、文の構成を調べて中間言語を生成する。生成された中間言語は最適化された後、目的言語に変換される。

ⓒ　字句解析

字句は構文を構成する最小単位である。バッファに読み込まれた文字列は、字句またはトークンの生成規則に従って、順次文法上の意味のある語、すなわち字句という一つ一つのかたまりとして切り出されていく。これが字句解析である。

字句解析では、正規表現を解析するプログラムが、ソースプログラムを読み込み、区切り記号に囲まれた構文要素を区別する。抽出された変数名、定数などは表にまとめられる。字句の生成規則、句の構文規則の表現には、正規表現が使用される。

ⓓ　字句の種類

㋐　名前

名前は変数や手続きなどにつけられた参照のための字句である。英字で始まる英数字で構成される。

㋑　予約語

予約語はコンパイラが特別の意味に使うために、あらかじめ定めた字句である。予約語には宣言や制御構造などの構文を示すための字句やコンパイラが提供する標準の定数や関数などがある。

㋒　定数

リテラルと呼ばれる文字列や数字列の字句を定数と呼ぶ。

㋓　区切り記号

字句を切り出す際の目安となる文字を区切り記号という。区切り記号は、スペース、カンマ、ピリオド、括弧、演算子などがある。

ⓔ　構文解析

㋐　構文解析とは

構文解析は、字句の連なりとして抽出したものを、文法に照らし合わせて合法な文であるかどうかを判断することである。文の種類には、宣言文、代入文、構造文などがある。構文解析は、原始プログラムのどの範囲が、式、文、ブロック、手続きなど構文上のどの概念と対比しているかを定め、その情報を木構造に組み立て、原始プログラムが構文規則に従っているかを検査する。構文解析を行うプログラムをパーサまたはシンタックスアナライザという。パーサによって、中間語を生成するために必要な解析木または構文木を作り出し、各種の表を次のフェーズのために準備する。

㋑　構文木

構文木は文や字句の構造を木構造で表現したものである。構文木に三つ組、四つ組、逆ポーランド記法などの表現法がある。四つ組は、(演算子、被演算子１、被演算子２、結果)の形式で、被演算子１と被演算子２に演算子を作用させたものが結果であることを表す。

ⓕ　意味解析

意味解析は変数名の未定義や二重定義、型の不一致など構文に含まれない制約を検査する。文字型として宣言された変数が算術式のなかに現れれば、意味上の誤りとして意味解析の段階に検出される。意味解析では、構文解析の解析結果を解釈して、中間語のコードを生成する。中間語の生成には、逆ポーランド記法や四つ組または三つ組などの手法を利用する。

ⓖ　最適化

最適化は、目的プログラムの実行時間や大きさが小さくなるように、プログラムを変更することである。最適化は構文解析と意味解析で得られた中間言語段階で行われる割合が大きいが、コード生成の段階でも行われる。

ⓗ　最適化の方法

㋐　定数畳込み

定数畳込みは、原始プログラムのなかで計算式を用いて求める定数がある場合、コンパイル時に計算して既知の定数値に置き換えておく処理である。例えば、変数ａ、ｂが与えられて、ｂ＝９－ａ／５の式を用いてｂの値を計算する場合、ａ＝30が定数として与えられると、ｂの計算式でコンパイル時に計算し、ｂ＝３を求め定数として使用する。プログラム実行時に計算する必要がなく、処理の高速化を図ることができる。

㋑　式の簡略化

同じ式を置き換える方法である。既に計算されている結果を利用し、再計算の無駄を省いたり、不要な計算や計算途中の値を削除したりする。定数による除算を逆数の乗算に置き換えたり、乗算をビットシフトと加算に置き換えるなどの方法がある。

㋒　ループの再編

❶　ループのなかで値の変わらない変数や式をループの外に移動する方法。
❷　多重ループを一重ループに置き換える方法。
❸　二つのループを一つにまとめる方法。
❹　ループを展開する方法。

㋓　レジスタ割付の再編

レジスタに余裕があれば、よく使われる値をレジスタに置いておく。

㋔　サブルーチンコールの組込

サブルーチンを呼び出す代わりに、サブルーチンの本体を組み込んでしまう方法。サブルーチンを呼び出す手続きが不要になる。

④　プログラムの編集、翻訳、実行まで

ⓐ　サービスプログラム

㋐　編集プログラム(テキストエディタ)

テキスト編集プログラムは、プログラムやデータ、文書などの文字列を対象に、効率よく入力したり、修正したりする編集プログラムで、基本機能としては、文字列の入力、文字列の削除、文字列の置換、文字列の複写、文字列の移動、ファイルの保管、ファイルの更新などがある。

図形エディタは、マウスなどのポインティングデバイスを使用して、図形や絵を作図するためのプログラムで、直線や長方形、円、楕円、自由曲線などを容易に描くことができる。基本機能としては、図形の作成、図形の移動・複写、図形の拡大・縮小、図形の修正、図形の変形、図形ファイルの保管・更新などがある。

㋑　ライブラリプログラム

❶　ライブラリの機能

編集プログラムで作成したプログラムを、磁気ディスクファイルや光ファイル上のライブラリに登録したり、不要になったプログラムをライブラリから削除する。

❷　ライブラリの種類

ライブラリには、次の種類がある。

ソースプログラムライブラリ：ソースプログラムを格納する

オブジェクトプログラムライブラリ：オブジェクトプログラムを格納する

ロードモジュールライブラリ：ロードモジュールを格納する

㋒　連係編集プログラム

㋓　ローダ

実行に先立ってロードモジュールを、主記憶装置内にローディングする機能を持つ。ロードモジュールは実行可能になっているが、プログラム内部のアドレスは相対アドレスである。この相対アドレスを記憶装置の実アドレスに変換しながらローディングし、ローディング完了後、記憶装置上に配置したプログラムの入口に制御を移し、実行を開始する。

㋔　追跡プログラム

プログラムの個々の命令が実行された順番に従って、命令を実行した結果や記憶場所の内容、レジスタの内容などを、時間の経過とともに出力するプログラムである。テスト支援ツールのトレーサやスナップショットダンプなどがある。

㋕　診断プログラム

コンピュータシステムに異常が発生した場合、その障害箇所を調査・分析し、表示・印刷出力するプログラムである。ハードウェアだけの診断ではなく、言語プロセッサ機能であるシンタックスエラーやセマンティックエラーに関するソフトウェア上の誤りの検出機能も含んでいる。

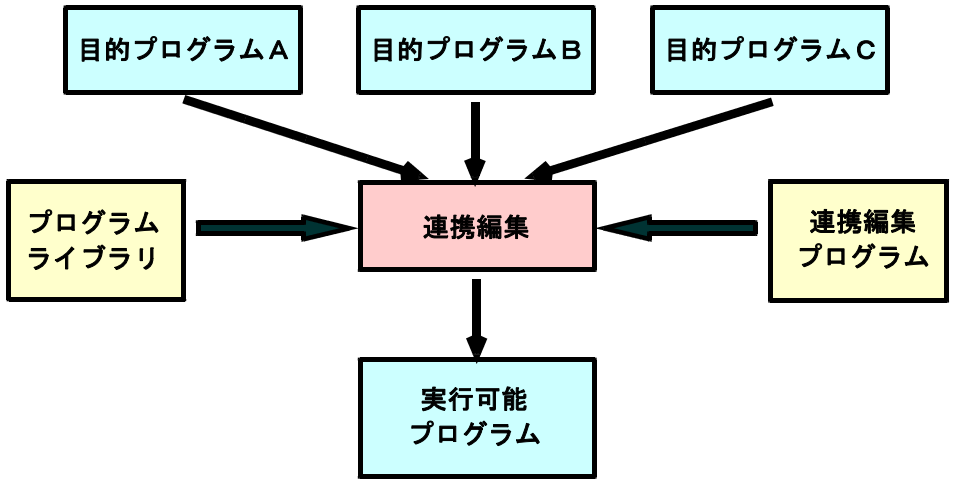
ⓑ　連係編集プログラム

㋐　連携編集プログラムとは

連携編集プログラム(リンケージエディタ、リンカ)は、コンパイラやアセンブラで出力された複数本のオブジェクトモジュールを結合して、１本のプログラムモジュールを作成するプログラムである。作成されたモジュールをロードモジュールという。結合とは、各プログラムモジュール間の呼出関係を把握して、ロードモジュールとしてまとめ上げることである。

㋑　連携編集プログラムの基本機能

連携編集プログラムの基本機能は、入力されたオブジェクトモジュールに連続したアドレスを割当て、各オブジェクトモジュール間の参照を結合し、ローダが読み込める形式の統合した１本のプログラムモジュールを作り上げることである。

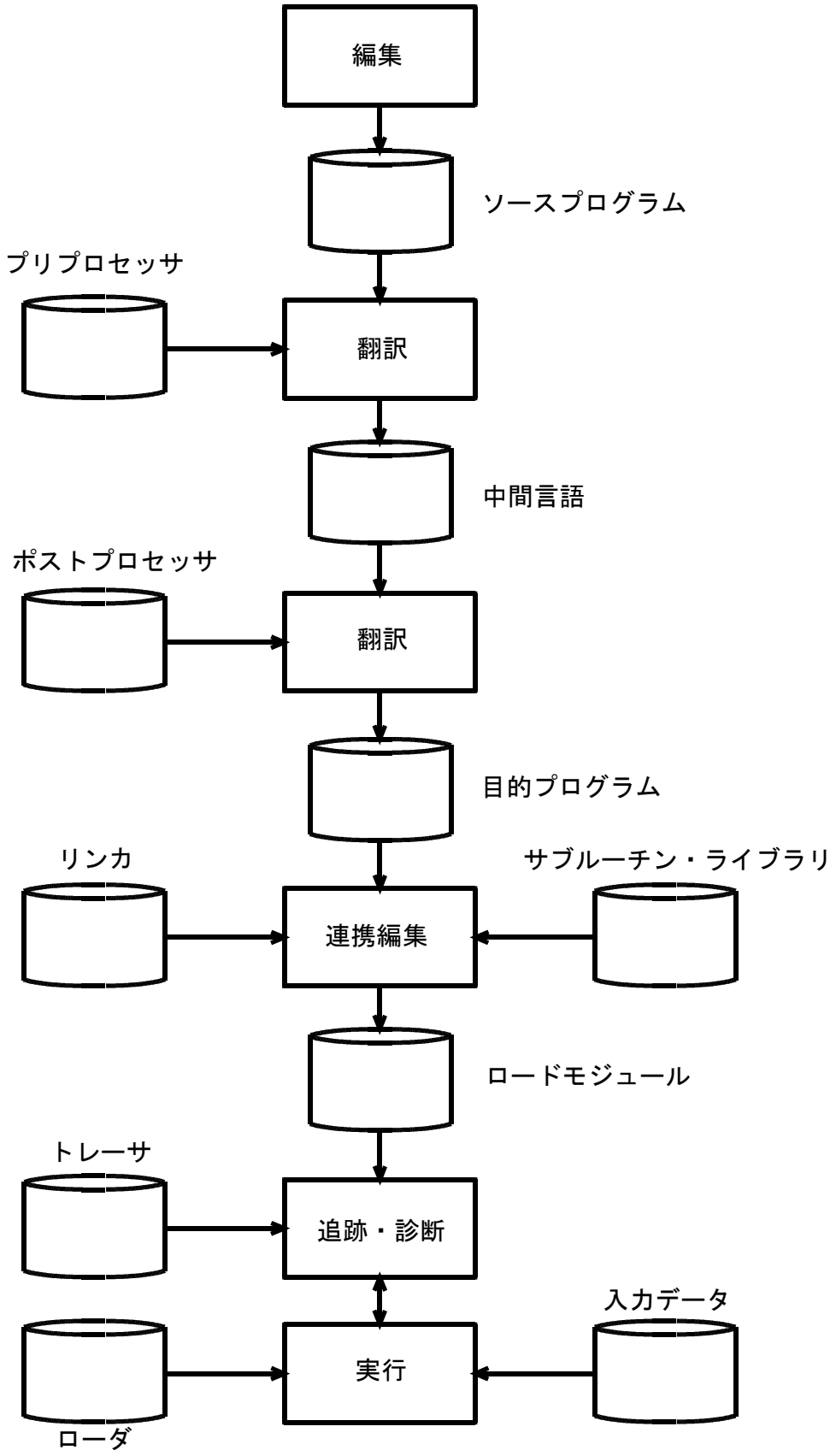


ⓒ　未解決アドレス処理

コンパイル時に決定できなかったアドレスは、記号名、型、開始アドレス、プログラムサイズなどが格納されている外部記憶辞書（ＥＳＤ）を利用して、別々に０番地の開始アドレスから生成された２本のオブジェクトモジュールを、アドレスが連続するように変更する。

マイクロコンピュータの場合は、プログラムの構造上、コードセグメントとデータセグメント、その他のセグメントに分け、プログラムを別々の領域に配置する。ジャンプ命令によるアドレス定数の修正の場合、再配置結合ディレクトリ（ＲＬＤ）を利用して、開始アドレスが変更されたセグメント内でのアドレスを、開始アドレスの移動分だけ修正する。ＲＬＤで覚えている定数の位置と加算または減算による修正法が行われる。

ⓓ　プログラムの編集から実行までのプロセス



⑤　プログラム構造

ⓐ　プログラム構造処理

㋐　静的リンキング

静的リンキングは、連携編集プログラムで結合することであり、ソフトウェアが必要としているｎ本のすべてのオブジェクトモジュールを結合する。その占有メモリ領域は、常駐プログラムの大きさに比例して大きくなる。

㋑　動的リンキング

動的リンキングは、記憶容量を少なくするために、すでにロードされている１本の実行形式のプログラムを複数のモジュールから実行時に初めて参照できるようにすることであり、記憶容量が大きくなりすぎる短所を解決する方法である。

㋒　オーバレイ方式

オーバレイ方式もリンカで作成される。実行時のプログラム領域の大きさに制限があるときに有効な手法で、割り当てられた固定の大きさの区画にプログラムが収納できるように配置する方法である。オーバレイ方式は、プログラムの排他関係を考慮して、いくつかのセグメントに分け、実行時にセグメントを制御する部分や、使用頻度の高いセグメントを常駐させ、その他のセグメントは実行時に必要になったときだけ、オーバレイ域にロードさせる。

ⓑ　リエントラント

リエントラントは、同じプログラムに対する複数の処理要求に同時に処理することができ、それぞれの処理要求に対して、正しい結果を返すことができる手続きであり、即時性が重要視されるオンライントランザクション処理を行うための機能である。複数の異なる処理要求に対して同時に応じることができ、正しい処理結果が得られるために、プログラム域とデータ域をメモリ上で分離して使用する。プログラム域はプロシージャの手続きを格納しているため、複数の処理要求で共有できる。プログラム域を複数の処理要求で共有することによって、メモリ空間を節約する。データ域の内容は個々の処理要求によって異なり、実行の過程で時々刻々変化していくために、複数の処理要求ごとに別々に管理する必要がある。プログラム域とデータ域はプログラム制御機能によって管理され、同時並行的に別々の処理を行うことができる。

ⓒ　プログラム域、作業域のメモリマップ

㋐　プログラム域はプログラム１個について一つだけ作る。
㋑　プログラム域は、常にメモリ上に常駐させる。
㋒　作業域は、プログラム１個について複数個作り、処理要求が終了すると解放する。
㋓　多重度は、プログラムＡに対する処理要求の同時に発生する頻度を想定して設定する。

ⓓ　リューザブル

リューザブルは再ローディングしないで、逐次再利用しても正しい結果を返すことのできる性格で、実行前に作業域を初期化する。そのルーチンで初期値を書き換えた場合、ルーチンの最後で元の値に戻してから復帰する必要がある。このプロシージャは並行処理はできないし、即時性にも向かない。

ⓔ　コード、データ、スタックの３領域

㋐　コード領域

コード領域は、コンピュータの命令の列であり、プログラムの実行中に書き込まれることのない領域である。１つのコード領域を複数のプログラムが共用できる。

㋑　データ領域

データ領域は、プログラムの実行中に使用するデータを置く、読み書き可能な領域である。

㋒　スタック領域

スタック領域は、関数の呼び出しにおいて、戻りアドレスやパラメータを格納するのに用いる。スタック領域は後入れ先出しの特徴を持った記憶領域で、スタックに情報を入れる操作をプッシュ、スタックからデータを取り出す操作をポップという。スタックは先頭アドレスを示すスタックポインタを持っている。スタックは、プロセス固有の領域として割り当てられる。スレッドをサポートするシステムでは、スレッドごとにスタックが割り当てられる。

⑥　手続・関数・サブルーチン

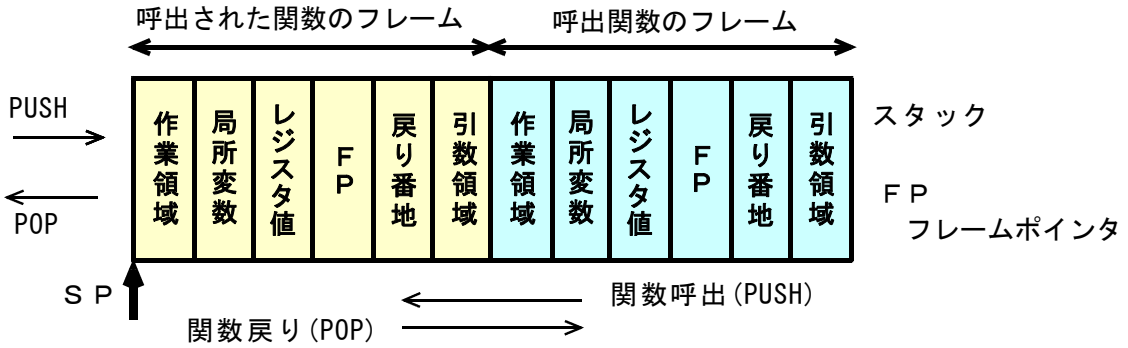
ⓐ　手続・関数とは

プログラムを開発する場合、システムの機能を分析し、段階的に詳細化し、幾つかのモジュールに分割する。全体から部分へプログラミングしていく考え方を構造化プログラミングという。構造化プログラムの言語で、プログラムを機能部分に分割して独立性を持たせる場合には、プログラムの中に手続あるいは関数と呼ばれるブロックを定義する。サブルーチンは、使用頻度の高い特定の独立した機能をもち、メインプログラムの中に組み込むことができるモジュールの単位である。一つのプログラムで何度も使用でき、複数のプログラムで使用できる。

ⓑ　プログラムの構造化

プログラムの流れをいくつかの意味を持つモジュールの集まりとして考え、個々のモジュールをメインプログラムと切り離して、小さな独立したプログラムとして定義する。この考え方を用いると、プログラム開発を容易にし、プログラムの保守や信頼性を高めることができる。

ⓒ　関数func（ａ，　ｂ）の呼出手順



㋐　引数ｂ、ａの順にスタックにプッシュする。

㋑　関数の呼び出し命令を実行する場合、関数の戻りアドレスをスタックにプッシュし、呼出関数にジャンプする。

㋒　呼出関数のフレームポインタをスタックにブッシュする。

㋓　呼出関数のレジスタの退避を行う。

㋔　関数の局所変数の領域をスタック上に確保し、指定があれば初期化を行う。

㋕　スタック上に呼出された関数の作業領域を確保する。

ⓓ　スタックフレーム

１回の関数呼び出しに対応して確保される管理情報やパラメータ、局所変数などを格納するための連続領域をフレームという。ＣやPASCALなどのプログラミング言語では、関数を呼び出すときにフレームがスタック上に確保され、復帰するときにスタックから取り除かれる。このようなフレームをスタックフレームという。

スタックフレーム内の情報を効率的に参照できるように、フレーム内の特定位置を指すベースレジスタが用いられる。これをフレームポインタという。フレームポインタはスタックフレームが確保されたときに設定し直され、それまで実行していた関数のスタックフレームの内容がスタックにプッシュされる。これらの操作は呼び出した関数の先頭で汎用レジスタの退避に先駆けて行われる。呼び出された関数の実行が終了すると、フレームポインタをもとに、スタックポインタをCALL命令を実行した直後の位置にセットし、フレームポインタの値を回復する。

サブルーチン復帰命令を実行すると、復帰アドレスをスタックからポップし、プログラムカウンタにセットする。呼び出し側では、パラメータを一度にポップして、スタックポインタを関数呼び出しの前の位置に戻す。

⑦　可視性と有効範囲

ⓐ　可視性とは

プログラムや関数内で宣言された変数が、どの範囲までプログラムを参照できるかを表した

ものである。有効範囲は参照できる範囲で、どこで変数が宣言されたかで決まる。

ⓑ　大域変数

大域変数は、変数がブロック外で宣言され、プログラムの全域で有効な変数であり、手続や関数からでも参照できる。

ⓒ　大域変数の特徴

㋐　どこからも参照できるので利用しやすい
㋑　プログラムの中で同じ名前の変数が定義されてはならない。
㋒　大域変数を参照しているモジュールやブロック間の結合が、不明確になる。
㋓　変数が変更になると参照している各モジュールやブロックに影響が生じる。
㋔　変更しにくいプログラムになる。

ⓓ　局所変数

局所変数は、ブロック内で宣言され、手続や関数の中だけで有効な変数であり、外側のブロックからは参照できない。同じ名前の変数が他のブロックにあっても関係ない。

ⓔ　C言語の変数

㋐　自動変数

自動変数は関数内で宣言し、関数内だけで有効な変数である。

㋑　静的変数

静的変数は関数内で宣言されれば関数内のみ、関数外で宣言されればコンパイル単位に有効となる。

㋒　レジスタ変数

レジスタ変数は関数内で宣言し、関数内だけで有効な変数である。

ⓕ　値渡しと参照渡し

㋐　値渡し

値渡しは、手続および関数の中で、パラメータを複写して、改めてそれを使用する。もとの値は変わることがなく、プログラムと手続、関数との独立性を保つことができる。

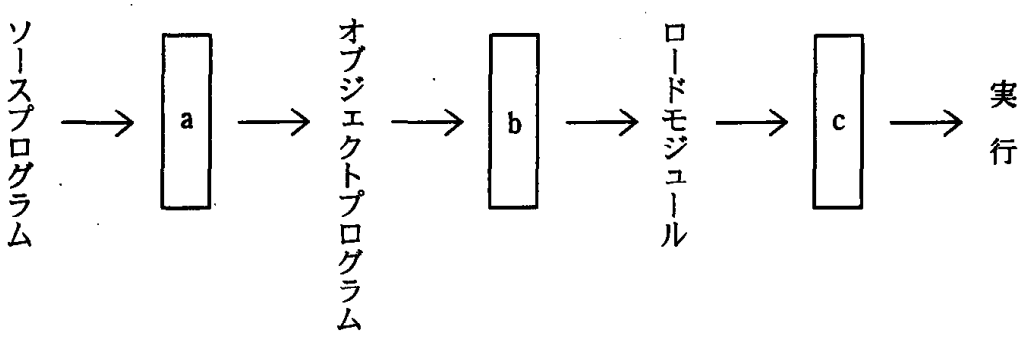
㋑　参照渡し

参照渡しは、手続および関数の中でパラメータのアドレスを知らせて、その同じデータを

参照または使用する。実引数と仮引数の場所は同じである。引数間でのデータのやりとりはない。パラメータのアドレスにしたがって、その値を参照したり、そこに値を格納することができる。他の手続や関数によって、予期しない間に、データを書き換えられてしまう危険性がある。モジュールの独立性を高めるためには、パラメータの値渡しが望ましい。

例題演習

ＣＯＢＯＬやＣでソースプログラムを作成した後，そのプログラムを実行するまでの手順として，ａ～ｃに入れるべきソフトウエアの適切な組合せはどれか。



	a	b	c
ア	コンパイラ	リンカ	ローダ
イ	コンパイラ	ローダ	リンカ
ウ	リンカ	コンパイラ	ローダ
エ	リンカ	ローダ	コンパイラ

解答解説

コンパイラの処理順序に関する問題である。

コンパイラは、ソースプログラムからオブジェクトプログラムを作成し、オブジェクトプログラムからリンカを使用してライブラリとリンクし、ロードモジュールを作成する。ローダは、データやプログラムをコンピュータに読み込むプログラムである。

ａはコンパイラ、ｂはリンカ、ｃはローダで、求める答えはアとなる。

例題演習

高水準言語で原始プログラムを作成した後、そのプログラムをコンパイル方式によって実行するまでの手順として、正しいものはどれか。

ア　原始プログラム作成→コンパイル→連携編集→ロード→実行
イ　原始プログラム作成→コンパイル→ロード→連携編集→実行
ウ　原始プログラム作成→連携編集→コンパイル→ロード→実行
エ　原始プログラム作成→連携編集→ロード→コンパイル→実行

解答解説

原始プログラム作成からロード、実行に至るプロセスに関する問題である。

正しい手順は、原始プログラムの作成→コンパイル→連携編集→ロード→実行の順に行われる。求める答えはアとなる。

例題演習

言語プロセッサに関する記述のうち、正しいものはどれか。

ア　アセンブラは、ある処理系用に書かれた原始プログラムを、他の処理系用の原始プログラムに変換する。

イ　インタプリタは、他のコンピュータ用のプログラムを解読し、実行するマイクロプログラムである。

ウ　ジェネレータは、入力・処理・出力に関するいろいろな条件をパラメタで記述し、問題の処理目的に応じたプログラムを生成する。

エ　トランスレータは、高水準言語で書かれたプログラムを、解釈しながら実行する。

解答解説

言語プロセッサに関する問題である。

ジェネレータは、条件をパラメータで記述し、問題の処理目的に応じたプログラムを自動的に生成することである。

アはトランスレータの内容であり、アセンブラではない。

イはシミュレータであり、インタプリタではない。

ウのジェネレータの記述のみが正しい内容である。求める答えはウとなる。

エはインタプリタに関する記述で、トランスレータではない。

例題演習

言語処理のコンパイラ、インタプリタに関して、正しい記述はどれか。

ア　インタプリタは、ＣＯＢＯＬやＦＯＲＴＲＡＮで利用されることが多い。

イ　プログラムの実行速度という点では、コンパイラを利用したときに比べてインタプリタを利用した方が速い。

ウ　インタプリタを使うと、プログラムが途中までしか完成していない場合でも、そこまでの部分を実行させることができる。

エ　コンパイラを使うと、ソースプログラムの命令文を一つずつ翻訳実行するので、１ステップずつデバッグするのに都合がよい。

解答解説

コンパイラ、インタプリタに関する問題である。

アのＣＯＢＯＬやＦＯＲＴＲＡＮはコンパイラである。

イのインタプリタの実行速度はコンパイラより遅い。

ウの記述は、インタプリタに関する内容である。求める答えはウとなる。

エの記述は、インタプリタに関するもので、コンパイラではない。

例題演習

コンパイラについて記述したものはどれか。

ア　原始プログラムを解釈し実行する。
イ　構文解析、意味解析を行い、コードを生成する。
ウ　実行するアドレスに合わせて、プログラムを再配置する。
エ　ロードモジュールを編集する。

解答解説

コンパイラに関する問題である。

アの原始プログラムを解釈し実行するは、インタプリタに関する記述である。

イの構文解析、意味解析を行い、コードを生成するは、コンパイラに関する記述であり、求める答えである。

ウの実行するアドレスに合わせたプログラムの再配置は、仮の記憶領域に割り付けられているプログラムの格納位置を、実行に必要な記憶領域の格納位置に再設定することでる。連係編集プログラムまたはロードモジュールプログラムの機能である。。

エのロードモジュールの編集は、オブジェクトモジュールを幾つか組み合わせて実行可能プログラムにする連携編集プログラムである。

例題演習

次の文はある二つの言辞処理系について記述したものである。Ｂと比べたＡの利点を記述しているものはどれか。

Ａ：高水準言辞で作成されたプログラムを，中間言語，アセンブラ言辞又は機械語で記述されたプログラムに翻訳する。
Ｂ：原始プログラム中の命令文を一文ずつ解釈し，実行する。

ア　処理の最適化が図れる。
イ　対話的な実行環境が構築できる。
ウ　デバッグ機能を組み込みやすい。
エ　プログラム作成とテストを並行してできる。

解答解説

言語プロセッサに関する問題である。

イ、ウ、エはインタプリタの特徴である。

アはコンパイラの特徴である。求める答えはアとなる。

例題演習

手続き形言語のコンパイラにおける処理を順に並べたものとして、正しいものはどれか。

ア　意味解析→構文解析→字句解析→最適化→コード生成
イ　意味解析→字句解析→構文解析→最適化→コード生成
ウ　字句解析→意味解析→構文解析→最適化→コード生成
エ　字句解析→構文解析→意味解析→最適化→コード生成

解答解説

コンパイラの処理順序に関する問題である。

手続き型言語のコンパイラの手順は、次の通りである。

字句解析→構文解析→意味解析→最適化→コード生成

の順に進められる。求める答えはエとなる。

例題演習

コンパイラで構文解析した結果の表現方法の一つに四つ組形式がある。

（演算子，被演算子１，被演算子２，結果）

この形式は，被演算子１と被演算子２に演算子を作用させたものが結果であることを表す。次の一連の四つ組は，どの式を構文解析した結果か。ここで，T_1，T_2，T_3は一時変数を表す。

（＊，Ｂ，Ｃ，T_1）
（／，T_1，Ｄ，T_2）
（＋，Ａ，T_2，T_3）

ア　Ａ＋Ｂ＊Ｃ／Ｄ　　イ　Ａ＋Ｂ＊Ｃ／T_2
ウ　Ｂ＊Ｃ＋Ａ／Ｄ　　エ　Ｂ＊Ｃ＋T_1／Ｄ

解答解説

四つ組形式に関する問題である。

（＊、Ｂ、Ｃ、T_1）はT_1＝Ｂ＊Ｃ、（／、T_1、Ｄ、T_2）はT_2＝T_1／Ｄ、（＋、Ａ、T_2、T_3）は　T_3＝Ａ＋T_2となる。

T_3＝Ａ＋T_1／Ｄ＝Ａ＋Ｂ＊Ｃ／Ｄとなる。求める答えはアとなる。

例題演習

プログラムの中で使用している外部関数を見つけ，未解決アドレスとして，次のステップに渡すものはどれか。

ア　コンパイラ　　イ　プリコンパイラ
ウ　リンケージエディタ　　エ　ローダ

解答解説

言語プロセッサ、サービスプログラムに関する問題である。

アのコンパイラは、プログラミング言語で記述されたプログラムを、コンピュータが理解できる機械語に一括変換するソフトウェアである。

イのプリコンパイラは、ソースプログラムの中のマクロ機能などを前処理して、コンパイラに対する単純化したソースプログラムモジュールを作成する。

ウのリンケージエディタは、コンパイラやアセンブラで出力された複数本のオブジェクトモジュールを結合して１本のプログラムモジュールを作成する。

エのローダは、ロードモジュールを再配置処理を行った上で、主メモリ上でロードするプログラムである。

リンケージエディタは、コンパイルに別々に０番地の開始アドレスから生成されていたオブジェクトモジュールにアドレスが連続するように変換する。外部関数を未解決アドレスとして次のステップに渡すのであるから、外部関数をコンパイラから次のステップであるリンケージエディタに渡すことになる。従って、求める答えはコンパイラになり、求める答えはアとなる。

例題演習

コンパイラの最適化に関する記述として、正しいものはどれか。

ア　オブジェクトコードを生成する代わりに、インタプリタ用の中間コードを生成する。
イ　コンパイルするコンピュータとは異なる機種で動作するオブジェクトコードを生成する。
ウ　プログラムが実行されたときに、制御が渡ったルーチン名やある時点での変数の内容を表示するようなオブジェクトコードを生成する。
エ　プログラムコードを解析して、実行時の処理効率がより高くなるようにオブジェクトコードを生成する。

解答解説

コンパイラの処理における最適化の問題である。

最適化とは、実行速度を速くしたり、小さなオブジェクトプログラムを作成するためにコンパイル時に行われる手段の一つで、実行速度を速める場合には、定数の計算を事前に行ったり、ループ内で変化しない演算式をループの外に出したりして最適化を行う。

アの中間コードはJavaマシン上で動作するバイトコードを生成する現象で、最適化ではない。イはクロスコンパイラ、ウはクロスリファランス、エが最適化である。求める答えはエとなる。

例題演習

コンパイラによる最適化の主な目的はどれか。

ア　プログラムの実行時間を短縮する。
イ　プログラムのデバッグを容易にする。
ウ　プログラムの保守性を改善する。
エ　目的プログラムを生成する時間を短縮する。

解答解説

コンパイラの処理における最適化の問題である。

最適化とは、実行速度を速くしたり、小さなオブジェクトプログラムを作成するためにコンパイル時に行われる手段の一つで、実行速度を速める場合には、定数の計算を事前に行ったり、ループ内で変化しない演算式をループの外に出したりして最適化を行う。

最適化の目的はプログラムの実行時間の短縮である。求める答えはアとなる。

例題演習

次の一連の３アドレス命令で得られる結果ｘを表す式はどれか。ここで，３アドレス命令では，三つのオペランドを用いた命令“ｃ＝ａ　ｏｐ　ｂ”を“ｏｐ（ａ，　ｂ，　ｃ）”として表記する。ｏｐは一つの演算子を表し，結果ｘを表す式においては優先順位の高い順に＊，／，＋，－とする。

／（ｃ，　ｄ，　ｗ１）
＋（ｂ，　ｗ１，　ｗ２）
／（ｅ，　ｆ，　ｗ３）
－（ｗ３，　ｇ，　ｗ４）
＊（ｗ２，　ｗ４，　ｘ）

ア　ｂ＋ｃ／ｄ＊ｅ／ｆ－ｇ　　イ　ｂ＋ｃ／ｄ＊（ｅ／ｆ－ｇ）
ウ　（ｂ＋ｃ／ｄ）＊ｅ／ｆ－ｇ　　エ　（ｂ＋ｃ／ｄ）＊（ｅ／ｆ－ｇ）

解答解説

四つ組みに関する問題である。

ｏｐ（ａ，ｂ，ｃ）は、ｃ＝ａ　ｏｐ　ｂを意味する表記であるから、与えられた四つ組みの表記を順次実行すると次のようになる。

／（ｃ，ｄ，ｗ１）　　ｗ１＝ｃ／ｄ
＋（ｂ，ｗ１，ｗ２）　　ｗ２＝ｂ＋ｗ１＝ｂ＋ｃ／ｄ
／（ｅ，ｆ，ｗ３）　　ｗ３＝ｅ／ｆ
－（ｗ３，ｇ，ｗ４）　　ｗ４＝ｗ３－ｇ＝ｅ／ｆ－ｇ
＊（ｗ２，ｗ４，ｘ）　　ｘ＝ｗ２＊ｗ４＝（ｂ＋ｃ／ｄ）＊（ｅ／ｆ－ｇ）

求める答えはエとなる。

例題演習

連携編集プログラムに関して、正しい記述はどれか。

ア　作成したプログラムをライブラリに登録する。
イ　実行に先立ってロードモジュールを主記憶に移す。
ウ　プログラムの構文や意味の誤りを検出する。
エ　目的モジュールやロードモジュールを組み合わせて、ロードモジュールを作成する。

解答解説

連係編集プログラムに関する問題である。

連係編集プログラムは、オブジェクトプログラムとプログラムライブラリから呼び出したサブルーチンを結合させ、実行可能なプログラムを作るプログラムである。複数本のコンパイル単位のモジュールのオブジェクトコードを一つのプログラムに連結編集して結合する。

アはライブラリアン、イはローダ、ウはコンパイラ、エは連係編集プログラムで、求める答えはエとなる。

例題演習

インタプリタの説明として、適切なものはどれか。

ア　アセンブラ言語で書かれた原始プログラムを、機械語のプログラムに翻訳するプログラムである。

イ　原始プログラムを、１文ずつ解析して実行するプログラムである。

ウ　高水準言語で書かれた原始プログラムを、機械語のプログラムに翻訳してロードモジュールを作るプログラムである。

エ　指定されたパラメータから、処理の目的に応じたプログラムを自動的に生成するプログラムである。

解答解説

インタプリタに関する問題である。

インタプリタはソースプログラムを１行ずつ解釈して機械語に翻訳しながら実行するソフトウェアで、プログラムの命令を逐次翻訳して実行していくため、実行速度は遅いが、文法エラーやプログラムの修正を対話的に行える特徴がある。インタプリタは、１行ごとに翻訳実行し、作成途中でもそれまでの部分を実行でき、変更が容易である。速度はコンパイラのほうが速い。

アはアセンブラ、イがインタプリタ、ウはコンパイラと連携編集プログラム、エはジェネレータである。求める答えはイとなる。

例題演習

プリコンパイラの説明として，適切なものはどれか。

ア　ある言語でコーディングされたプログラムを，別の言語のプログラムに変換するプログラムである。

イ　あるコンピュータ上で実行されるオブジェクトプログラムを，それとはアーキテクチャが異なるコンピュータ上で生成するコンパイラである.

ウ　高水準言語でコーディングされたソースプログラムを．オブジェクトプログラムに変換するプログラムである。

エ　高水準言語に付加的に定義された機能と文法に従ってコーディングされたプログラムを，元の高水準言語だけを使用したプログラムに変換するプログラムである。

解答解説

プリコンパイラに関する問題である。

プリコンパイラはコンパイルに先立って、コンパイラ言語に付加された特殊な機能を通常のコンパイラ言語に変換する。

アはトランスレータ、イはクロスコンパイラ、ウはコンパイラ、エはプリコンパイラで、求める答えはエとなる。

例題演習

動的リンクライブラリ（ＤＬＬ）に関する正しい記述はどれか。

ア　コンパイル時に、コンパイラによって組み込まれる。
イ　コンパイルの前に、プリコンパイラによって生成される。
ウ　実行時に、オペレーティングシステムによって連携される。
エ　ロードモジュール作成時に、連携編集プログラムによって連携され組み込まれる。

解答解説

動的リンクライブラリに関する問題である。

アのコンパイラによる組み込みは、埋め込み方式のＳＱＬ文をプリコンパイラに通して、親言語のＣＡＬＬ文に変換し、そのソースプログラムを親言語のコンパイラに通して、ＣＡＬＬ文とモジュールを結合する場合などに利用される。

イのプリコンパイラによって生成されるのは、ソースプログラムの中のマクロ機能などの前処理に用いられる。

ウは動的リンク、エは静的リンクである。求める答えはウとなる。

例題演習

動的リンクライブラリ（ＤＬＬ）の特徴として，適切なものはどれか。

ア　アプリケーションがメモリにロードされるときに，同時にリンカによって組み込まれる。
イ　アプリケーションの実行中，必要になったときにＯＳによって連係される。
ウ　コンパイル時に，コンパイラによってアプリケーションに組み込まれる。
エ　コンパイルの前に，プリコンパイラによってアプリケーションに組み込まれる。

解答解説

動的リンクライブラリに関する問題である。

アは静的リンク、イは動的リンクである。求める答えはイとなる。

ウのコンパイラによる組み込みは、埋め込み方式のＳＱＬ文をプリコンパイラに通して、親言語のＣＡＬＬ文に変換し、そのソースプログラムを親言語のコンパイラに通して、ＣＡＬＬ文とモジュールを結合する場合などに利用される。

エのプリコンパイラによって生成されるのは、ソースプログラムの中のマクロ機能などの前処理に用いられる。

例題演習

動的リンキングの機能はどれか。

ア　ブログラム実行時に，共用ライブラリやシステムライブラリのモジュールをロードする。
イ　ブログラム実行時に，適切なアドレスに目的プログラムをロードする.
ウ　プログラム実行時に，読み込まれたページの論理アドレスを物理アドレスに変換する。
エ　プログラムの実行に先立って，複数の目的プログラムを連係編集（リンケージエディット）する。

解答解説

動的リンキングに関する問題である。

アは動的リンキング、イは静的ローディング、ウは動的ローディング、エは連携編集プログラムによるロードモジュールの作成である。求める答えはアとなる。

例題演習

プログラムを構成するモジュールの結合を，プログラムの実行時に行う方式はどれか。

ア　インタブリタ　　イ　オーバレイ
ウ　静的リンキング　　エ　動的リンキング

解答解説

動的リンキングに関する問題である。

アのインタプリタは、ソースプログラムを１行ずつ解釈して機械語に翻訳しながら実行するソフトウェアで、プログラムの命令を逐次翻訳して実行していくため、実行速度は遅いが、文法エラーやプログラムの修正を対話的に行える特徴がある。

イのオーバレイは、主記憶装置に読み込めない大きなプログラムを実行するときに複数のセグメントに分割して実行する方法である。常駐部のルートセグメントが、必要に応じて複数個に分割された排他関係にあるセグメントを交互に読み込んで実行する。

ウの静的リンキングは、コンパイラやアセンブラで出力された複数本のオブジェクトモジュールをリンケージエディタで結合して１本のプログラムモジュールを作成することである。

エの動的リンキングは、プログラムの実行中に別のプログラム・モジュールの機能が必要になったとき、そのプログラムをその場で結合して利用することである。求める答えはエとなる。

例題演習

言語プロセッサが動作するコンピュータとは別の機械語をもつコンピュータの目的プログラムを生成する言語プロセッサはどれか。

ア　クロスコンパイラ　　イ　コンパイラコンパイラ
ウ　プリプロセッサ　　エ　フロントエンドプロセッサ

解答解説

クロスコンパイラに関する問題である。

アのクロスコンパイラは、ほかのコンピュータで動くオブジェクトプログラムをつくるためのコンパイラである。求める答えはアとなる。

イのコンパイラコンパイラは、プログラミング言語の仕様を与えると、コンパイラプログラムを自動的につくるためのソフトウェアである。

ウのプリプロセッサは、プログラムの作成能力を上げるために、コンパイラで翻訳できない拡張した疑似命令を使用して作られた原始プログラムを、前もって解読可能な命令に置き換えるプログラムである。ソース・コード中に頻繁に出てくる数値を文字に置き換えておいたり、いつも必要なライブラリの初期化などの一連の手続きを別のファイルに用意しておいて、ソース・コードではそこを参照するようにしておくような機能に用いられる。C言語などのコンパイラで用いられ、コンパイル前にマクロの展開やヘッダファイルの挿入などの処理を行う。

エのフロントエンドプロセッサは、ホストコンピュータに先立って、通信制御などを行う処理装置やかな漢字変換を行うプログラムを示す場合に用いられる。

例題演習

クロスコンパイラに関する記述として、適切なものはどれか。

ア　開発対象のプログラムを実行するコンピュータとは異なる種類のコンピュータで動作する。
イ　原始言語と目的言語の定義からコンパイラを自動的に生成する。
ウ　コンパイラによる翻訳に先んじて、拡張原始言語を原始言語に展開する。
エ　プログラムを１ステップずつ読み込んで実行し、実行過程を追跡する。

解答解説

クロスコンパイラに関する問題である。

クロスコンパイラは他のコンピュータシステムで動作する機械語を生成するためのコンパイラである。あるコンピュータで動作するプログラムを作るときに、他のコンピュータの開発環境を用いてプログラムを作成する場合に使用する。動作速度が遅いコンピュータや完成していないコンピュータ、開発環境のない組込用コンピュータなどのソフトウェア開発に使用される。

アがクロスコンパイラ、イはコンパイラコンパイラ、ウはプリプロセッサ、エはトレーサである。求める答えはアとなる。

例題演習

異なる命令形式をもつコンピュータ用の目的プログラムを生成する言語処理プログラムはどれか。

ア　エミュレータ　　イ　クロスコンパイラ
ウ　ジエネレータ　　エ　シミュレータ

解答解説

クロスコンパイラに関する問題である。

アのエミュレータは、あるシステム上で、異なるシステム用に開発されたＯＳやアプリケーションを動作させるソフトウェアやハードウェアのことである。

イのクロスコンパイラは、あるコンピュータ上で高水準言語で書かれたプログラムを、そのコンピュータとは別のコンピュータのオブジェクトプログラムに翻訳するプログラムである。

ウのジェネレータは、非手続き型言語であるＲＰＧなどで作成されたプログラムを、機械語である目的プログラムに変換する翻訳処理プログラムである。

エのシミュレータは、ある計算機で実行可能な目的プログラムの命令を１つずつ解釈し、他の計算機の命令に変換して実行する場合に利用するプログラムである。シミュレータ実行時は、解釈ルーチンが主記憶に常駐するので領域を多くとり処理効率が悪く、処理速度も遅い。

異なる命令形式をもつコンピュータ用の目的プログラムを生成するのはクロスコンパイラであり、求める答えはイとなる。

例題演習

プログラミングツールに関する記述のうち，適切なものはどれか。

ア　デバッグ時にデータ構造の内容を確認するためのツールをインスベクタという。
イ　ブログラム単位の機能説明や定義を容易に探索するためのツールをトレーサという。
ウ　プログラム内又はプログラム間の実行経路を確認するためのツールをシミュレータという。
エ　プログラムのソースコードを編集するために，文字の挿入，削除，置換えなどの機能をもつツールをブラウザという。

解答解説

プログラミングツールに関する問題である。

アのインスペクタは、プログラムを実行してエラー検出やデータ構造の内容を確認するためのデバッギングツールで、プログラムを途中で中断し、トレース対象データの閲覧や更新を対話形式で処理する。求める答えはアとなる。

イのトレーサは、プログラムの実行時に制御の流れを追跡するときや実行過程を時系列的に１ステップずつモニタリングするときに使用する。

ウのシミュレータは、ある計算機で実行可能な目的プログラムの命令を１つずつ解釈し、他の計算機の命令に変換するプログラムである。

エのブラウザは、情報を閲覧するためのソフトウェアである。

例題演習

あるコンピュータの目的プログラムの命令を一つずつ解釈し、他のコンピュータの命令に変換してから実行するプログラムはどれか。

ア　シミュレータ　　イ　トランスレータ
ウ　ジェネレータ　　エ　クロスコンパイラ

解答解説

シミュレータに関する問題である。

アのシミュレータは、ある計算機で実行可能な目的プログラムの命令を１つずつ解釈し、他の計算機の命令に変換して実行する場合に利用するプログラムである。シミュレータの実行時は、解釈ルーチンが主記憶に常駐するので領域を多くとり処理効率が悪くなり、処理速度も遅い。求める答えはアとなる。

イのトランスレータは、ある機種用に開発された原始プログラムを他の機種用に変換するための言語で、機種切り換え時に利用する。

ウのジェネレータは、非手続き型言語であるＲＰＧなどで作成されたプログラムを、機械語である目的プログラムに変換する翻訳処理プログラムである。

エのクロスコンパイラは、あるコンピュータ上で高水準言語で書かれたプログラムを、そのコンピュータとは別のコンピュータのオブジェクトプログラムに翻訳するプログラムである。

例題演習

入力データの様式や処理内容などのパラメータを取り入れ、機械語のプログラムを作り出すのはどれか。

ア　ジェネレータ　　イ　アセンブラ
ウ　インタプリタ　　エ　コンパイラ

解答解説

ジェネレータに関する問題である。

アのジェネレータは、非手続き型言語であるＲＰＧなどで作成されたプログラムを、機械語である目的プログラムに変換する翻訳処理プログラムである。求める答えはアとなる。

イのアセンブラは、アセンブラ言語で記述された原始プログラムを目的プログラムに変換する翻訳プログラムで、原始プログラムの命令と出力する機械語の目的コードが１対１に対応しているので、翻訳作業は単純である。

ウのインタプリタは、原始プログラムを目的プログラムに変換せずに、そのまま実行する方式である。インタプリタは、解釈・実行を繰り返して行うのでプログラムが完成していなくても試験的に実行して処理結果を判断し、エラーがあればただちに修正することができる。

エのコンパイラは、高水準・手続き型言語で記述された原始プログラムを機械語の目的プログラムに翻訳するプログラムである。

例題演習

自分自身を直接的または間接的に呼びだして使用するサブルーチンをなんと呼ぶか。

ア　オーバレイサブルーチン　　イ　関数サブルーチン
ウ　再帰的サブルーチン　　エ　再入可能サブルーチン

解答解説

再帰プログラムに関する問題である。

アのオーバレイサブルーチンは、プログラムをセグメントという単位に分割して補助記憶装置に格納し、実行時に必要なセグメントを主記憶装置に格納する手続きである。

イの関数サブルーチンは、データを入力すると、一定の規則に従って計算結果を出力する仕組みである。主プログラムの中で繰り返し使用される処理を関数として宣言する。

ウの再帰的サブルーチンは、プログラムの中に記述された命令で、そのプログラム自身を呼び出すことができるプログラムである。求める答えはウとなる。

エの再入可能サブルーチンは、同じプログラムに対する複数の処理要求に同時に処理することができ、それぞれの処理要求に対して、正しい結果を返すことができるプログラムまたは手続きである。

例題演習

再帰的プログラムの特徴として，最も適切なものはどれか。

ア　一度実行した後，ロードし直さずに再び実行を繰り返しても，正しい結果が得られる。
イ　実行中に自分自身を呼び出すことができる。
ウ　主記憶上のどこのアドレスに配置しても，実行することができる。
エ　同時に複数のタスクが共有して実行しても，正しい結果が得られる。

解答解説

再帰的プログラムの特徴に関する問題である。

リカーシブ(再帰)は、プログラムの中から自分自身を呼び出すことを言う。自分自身を定義するのに自分自身よりも１次低い集合を用いる。その部分集合はより低次の部分定義を用いて定義することを繰り返して表現する。手続きの内部で再び自分自身を呼び出すことを再帰呼出という。

アはリューザブル、イはリカーシブ、ウはリロケータブル、エはリエントラントである。求める答えはイとなる。

例題演習

再帰呼出しの説明はどれか。

ア　あらかじめ決められた順番ではなく，起きた事象に応じた処理を行うこと
イ　関数の中で自分自身を用いた処理を行うこと
ウ　処理が終了した関数をメモリから消去せず，必要になったとき再び用いること
エ　処理に失敗したときに，その処理を呼び出す直前の状態に戻すこと

解答解説

再帰呼び出しに関する問題である。

再帰呼び出しは、関数の処理中に自分自身の関数を呼び出して処理を進めることである。再

帰呼び出しでは、必ず、再帰呼び出しからの脱出条件が存在し、再帰からの脱出が行われると、逐次、元の処理に戻っていく。

アはフィードバック制御、イは再帰呼び出し、ウは静的関数、エは復帰である。求める答えはイとなる。

例題演習

複数のプロセスから同時に呼び出されたときに，互いに干渉することなく並行して処理することができるプログラムの性質を表すものはどれか。

ア　リエントラント　　イ　リカーシブ
ウ　リユーザブル　　エ　リロケータブル

解答解説

プログラムの性質に関する問題である。問題の対象はリエントラントである。

アのリエントラントは、手続きだけを記憶したプログラム域と実行によって内容が変化していく部分を記憶するデータ域を分離し、複数の実行要求に対して同時に処理することのでき、それぞれの処理要求に対して正しい結果を返すことができるプログラムまたは手続きである。。求める答えはアとなる。

イのリカーシブは、プログラムの記述などで、関数や手続きを、その関数や手続きを使って定義することである。また、自分自身を直接または間接的に呼び出すことを再帰呼出という。

ウのリューザブルは、再ローディングしないで、逐次再利用しても正しい結果を返すことができる特徴を有するプログラムのことである。

エのリロケータブルは、記憶装置を効率的に使用するために、プログラムをメモリ内で移動させることである。プログラムの実行中に行うものを動的再配置、そうでないものを静的再配置という。

例題演習

プログラムの構造に関する記述のうち、正しいものはどれか。

ア　再帰的処理のためには、実行途中の状態をＦＩＦＯ方式で記録し、制御する必要がある。
イ　再入可能プログラムを実現するためには、プログラムを手続部分とデータ部分に分割して、データ部分をプロセスごとにもつことが必要である。
ウ　逐次再使用可能なプログラムは、再入可能でもある。
エ　複数のプロセスで同時に実行できるようにしたプログラムは、再帰的であるという。

解答解説

プログラムの構造に関する問題である。

アの再帰的処理の特徴は、スタックを利用するＬＩＦＯ方式であり、ＦＩＦＯ方式ではない。

イの再入可能プログラムは、手続部分とデータ部分に分割して、データ部分をプロセス毎に持つことが必要である。正しい記述である。求める答えはイとなる。

ウの逐次再使用可能なプログラムは、データ部分のプロセス毎の管理が可能になっていないため再入可能では使用できない。

エは、再入可能であって、再帰的ではない。

例題演習

再入可能(リエントラント)プログラムの説明として、最も適切なものはどれか。

ア　一度実行した後、ロードし直さずに再び実行を繰り返しても、正しい結果が得られる。
イ　実記憶上のどこのアドレスに配置しても実行することが可能である。
ウ　複数のセグメントに分割されて、セグメント単位にロードして実行することが可能である。
エ　複数のタスクで並行して実行しても、正しい結果が得られる。

解答解説

リエントラントに関する問題である。

リエントラントは、手続きだけを記憶したプログラム域と実行によって内容が変化していく部分を記憶するデータ域を分離し、複数の実行要求に対して同時に処理することができ、それぞれの処理要求に対して正しい結果を返すことができるプログラムまたは手続きである。

アはリューザブル、イはリロケータブル、ウはオーバレイ、エはリエントラントで、求める答えはエとなる。

例題演習

処理が終了していないプログラムが，別のプログラムから再度呼び出されることがある。このプログラムが正しく実行されるために備えるべき性質はどれか。

ア　再帰的(リカーシブ)　　イ　再使用可能(リユーザブル)
ウ　再入可能(リエントラント)　　エ　再配置可能(リロケータブル)

解答解説

プログラムの性質に関する問題である。

アの再帰的(リカーシブ)は、プログラムの記述などで、関数や手続きを、その関数や手続きを使って定義することである。自分自身を直接または間接的に呼び出すことを再帰呼出という。

イの再使用可能(リューザブル)は、再ローディングしないで、逐次再利用しても正しい結果を返すことができる特徴を有するプログラムのことである。

ウの再使用可能(リエントラント)は、手続きだけを記憶したプログラム域と実行によって内容が変化していく部分を記憶するデータ域を分離し、複数の実行要求に対して同時に処理することができ、それぞれの処理要求に対して正しい結果を返すことができるプログラムまたは手続きである。求める答えはウとなる。

エの再配置可能(リロケータブル)は、記憶装置を効率的に使用するために、プログラムをメモリ内で移動させることである。プログラムの実行中に行うものを動的再配置、そうでないものを静的再配置という。

例題演習

次の記述中の［　　］に入れるべき適切な字句の組合せはどれか。

主記憶にロードされた一つの［　a　］を複数の［　b　］で共用して並行に実行するためには、その［　a　］が［　c　］であることが必要である。

	a	b	c
ア	サブルーチン	タスク	再帰的
イ	タスク	サブルーチン	再帰的
ウ	タスク	プログラム	再入可能
エ	プログラム	タスク	再入可能

解答解説

リエントラントに関する問題である。

ａはプログラム、ｂはタスク、ｃは再入可能である。求める答えはエとなる。

例題演習

次のプログラムを実行したときの結果はどれか。

ここで，仮引数ｘは値呼出し(call by value)，ｙは参照呼出し(call by reference)とする。

```
メインプログラム                サブプログラム sub(ｘ，ｙ)
  ａ＝３；                        ｘ＝ｘ＋ｙ；
  ｂ＝２；                        ｙ＝ｘ＋ｙ；
  sub(ａ，ｂ)；                   return;
```

ア　ａ＝３，ｂ＝２　　イ　ａ＝３，ｂ＝７

ウ　ａ＝５，ｂ＝２　　エ　ａ＝５，ｂ＝７

解答解説

値呼び出し、参照呼び出しに関する問題である。

値渡しは、プログラムにおける主プログラムと副プログラム間のデータの受渡し方式の一つで、プログラムにデータ(引数)を与えて処理を行わせることである。与えられたデータが変化しても元のデータに影響を与えない。参照渡しは、メイン・プログラムとサブルーチンの間でデータを受け渡す方式の一つで、仮引数である変数が実行中に変化すると、自動的に実引数である変数も変化する。多くのプログラム言語では、サブルーチンに対して引数を、引数のアドレスで引き渡す。サブルーチン側でこの引数を操作する場合には、そのアドレスを参照することにより行う。

ｘは値呼び出しであり、ｙは参照呼び出しであるから、次のように計算される。

ｘ＝ｘ＋ｙでは、ｘ＝３、ｙ＝２で、計算後、ｘ＝５、ｙ＝２である。

ｙ＝ｘ＋ｙでは、ｘ＝５、ｙ＝２で、計算後、ｘ＝５、ｙ＝７で、サブプログラムからのリターン値はないため、メインプログラムのａ、ｂの値は、ａは変化しない、ｂは参照呼び出しのためｙが求まった段階で修正されて、ｂ＝７となる。

従って、ａ＝３、ｂ＝７となり、求める答えはイとなる。

例題演習

プログラム言語における関数呼出し時の引数の性質のうち，適切なものはどれか。

ア　値呼出しでは，仮引数の値を変えると実引数の値も変わる。
イ　実引数から仮引数に情報を渡す方法として，値呼出し，参照呼出しなどがある。
ウ　実引数は変数だけであるが，仮引数は変数でも定数でもよい。
エ　実引数は呼び出される関数の中だけで有効であるが，仮引数は関数の呼出し側でも有効である。

解答解説

関数呼び出し時の引数の性質に関する問題である。

引数はプログラムが関数を呼び出すとき、関数で引用する呼び出し側の情報である。呼び出し側と関数は引数を使用して情報の受け渡しを行う。呼び出し側で渡す情報を実引数、関数側で受け取る情報を仮引数という。

値渡しの場合、実引数の値は異なるメモリ領域に複写される。複写された変数は呼び出された関数の中だけで使用され、関数外には反映されない。参照渡しの場合、関数の引数にポインタを渡す。関数に渡されたポインタは元のメモリ領域を参照するので、呼び出された関数内から直接アクセスすることができる。従って、関数内で変数の値が変更されると元の値も直接変更されることになる。

アの値渡しの場合、仮引数の値が変わっても、実引数の値は変わらない。

イの実引数から仮引数に情報を渡す方法には値呼び出しと参照呼び出しの方法がある。求める答えはイとなる。

ウの実引数も変数も定数もある。

エの仮引数は値呼び出しの場合、関数外では有効でない。

例題演習

サブルーチンへの引数の渡し方のうち，変数を引数として渡しても，サブルーチンの実行後に変数の値が変更されないことが保証されているものはどれか。

ア　値呼出し　　イ　結果呼出し　　ウ　参照呼出し　　エ　名前呼出し

解答解説

関数呼び出し時の引数の性質に関する問題である。

引数はプログラムが関数を呼び出すとき、関数で引用する呼び出し側の情報である。呼び出し側と関数は引数を使用して情報の受け渡しを行う。呼び出し側で渡す情報を実引数、関数側

で受け取る情報を仮引数という。

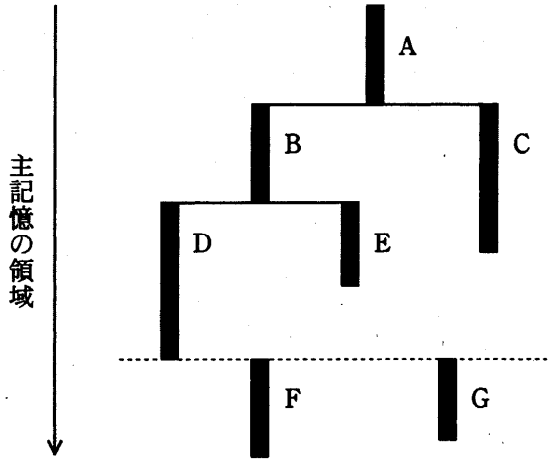
値渡しの場合、実引数の値は異なるメモリ領域に複写される。複写された変数は呼び出された関数の中だけで使用され、関数外には反映されない。参照渡しの場合、関数の引数にポインタを渡す。関数に渡されたポインタは元のメモリ領域を参照するので、呼び出された関数内から直接アクセスすることができる。従って、関数内で変数の値が変更されると元の値も直接変更されることになる。

サブルーチンの実行後に変数の値が変更されないのは値渡しの場合である。求める答えはアとなる。

例題演習

モジュールＡ～Ｅ，共通ルーチンＦ，Ｇで構成されるプログラムを図のようなオーバレイ構造にしたとき，参照が許されるのはどれか。図は，例えばモジュールＤの実行時には，主記憶にＡ、Ｂ，Ｄがロードされることを表している。また，“Ｘ→Ｙ”はモジュールＸからＹ内の領域をアクセスすることを表す。

ア　Ｂ→Ｃ
イ　Ｃ→Ｄ
ウ　Ｆ→Ａ
エ　Ｇ→Ｆ



解答解説

オーバレイに関する問題である。

オーバレイ方式は、主記憶容量より大きなプログラムを実行できるようにする方式で、プログラムをセグメントという単位に分割して補助記憶装置に格納し、実行時には実行に必要なセグメントをその都度補助記憶装置から主記憶装置へローディングする方式である。セグメント大きさの決定、主記憶装置への割当、補助記憶装置から主記憶装置へのローディングの指示はプログラム内で決定する。主記憶装置と補助記憶装置との間の情報転送はプログラムが分割されたセグメント単位に行われる。オーバレイでローディングされるセグメントは、同時に主記憶装置上に存在する必要のない排他的なプログラム部分である。共通して使用されるセグメントは主記憶装置内に常駐する。

アのＢとＣ、イのＣとＤ、エのＦとＧは排他的関係にあり、主記憶に同時に存在しないため参照することはできない。ウのＦとＡは同時に存在し、参照が可能である。求める答えはウとなる。

例題演習

主プログラムMainと副プログラムSubXからなる図のプログラムを実行した後の，変数Ａ，Ｂの値の組合せとして，正しいものはどれか。ここで，プログラム中の[]の部分は，コードの代わりにその内容を記述したものである。

```
Main
  [変数Ａの宣言]
  [変数Ｂの宣言]
  Ａ ＝ 1
  Ｂ ＝ 2
  Call  SubX （Ａ，Ｂ）
End
```

```
SubX（参照渡しの仮引数Ｃ，値渡しの仮引数Ｄ）
  [変数Ｅの宣言]
  Ｅ ＝ Ｃ
  Ｃ ＝ Ｄ
  Ｄ ＝ Ｅ
End Sub
```

	Ａ	Ｂ
ア	1	1
イ	1	2
ウ	2	1
エ	2	2

解答解説

参照渡し、値渡しに関する問題である。

参照渡しは、ある変数のメモリ上のアドレスを得て、その変数の値にアクセスすることである。結果の格納もアドレスの位置に格納される。値渡しは、値をコピーして渡すことである。処理した結果はコピーされた変数の修正になる。コピー前の領域の値は変わらない。

SubXの関数の引数の条件から、Ａは参照渡しで引数Ｃに渡される。Ｂは値渡しで引数Ｄに渡される。Ｃの値はＥに代入され、Ｄの値はＣに代入され、Ａに戻される。

Ｄ＝Ｂ＝２であるから、Ｃ＝２、Ａ＝２となる。Ｂの値は値渡しであり、SubX関数の処理と関係しないため、Ｂ＝２となる。従って、答えはＡ＝２、Ｂ＝２となり、求める答えはエとなる。

txt040313 プログラムテスト

① テスト

ⓐ テストの目的

テストの目的は作成したソフトウェア製品の品質を評価することである。開発したシステムやモジュールが設計仕様書どおりに動作するかなど、ソフトウェアの完成度を検証する一連の作業である。開発者とユーザがそれぞれの観点からテストを実施し、システムやモジュールの潜在的な欠陥を発見することでもある。計画、設計、製作の段階はトップダウンアプローチで進められるが、テスト段階以降はボトムアップアプローチの考え方が用いられる。

ⓑ テスト時の主要評価項目

㋐ 信頼性は要求仕様を満足し、ソフトウェアにエラーがなく、規則違反の使用に対して堅固であることである。

㋑ 操作性は入力しやすいこと、入出力情報やメッセージの内容が読みやすく、理解しやすいことである。

㋒ 性能はソフトウェアの処理能力や所要記憶容量が目標値であることである。

㋓ 保守性は修正・改造が容易であり、拡張性があることである。

② テストの種類

ⓐ テストの種類とその内容

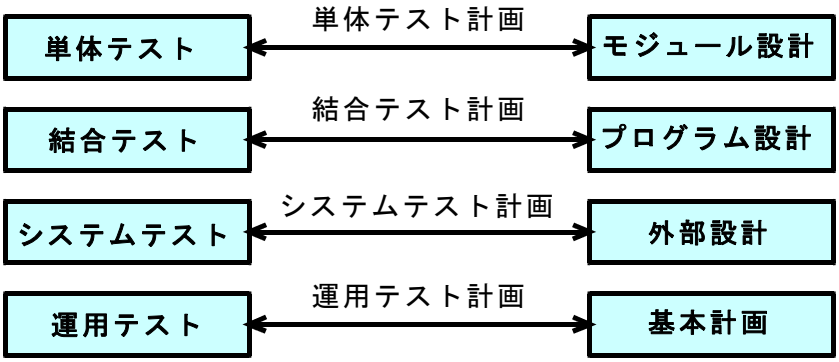
- 単体テスト（モジュールテスト：モジュール内のバグを見つけるテスト）
- 結合テスト
  - モジュール間インタフェーステスト
  - プログラムの入出力を含む一貫テスト
- システムテスト
  - プログラム間の結合テスト
  - サブシステム間の結合テスト
  - サブシステムおよび全体システムの入出力を含む一貫テスト
  - 例外事項テスト
  - 障害テスト
  - 処理速度テスト
- 運用テスト(承認テスト)

ⓑ　テストの順序

テストは、単体テスト、結合テスト、システムテスト、承認テスト、運用テストの順に進められる。単体テストからシステムテストはシステム開発者が主体で進める。承認テスト、運用テストはシステムのユーザが主体で進めるテストになる。

ⓒ　テスト計画の作成時期とテストの関係

テスト計画はテストの種類に応じて、計画、設計段階に作成される。

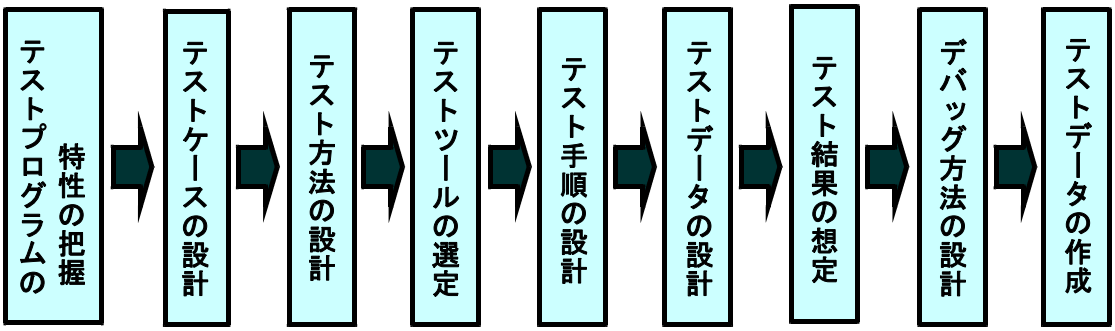


③　テスト設計

ⓐ　テスト設計とは

テストは限られた時間とコストの範囲内で実行する必要がある。単体テスト、結合テスト、システムテスト、運用テストなどのテストの種類に応じて、テストの目的、テストの方法を明確にし、テスト対象のプログラムの特性を吟味して、適切なテスト技法、テストツール、テストデータを選択・設計しなければならない。テストの種類やプログラムの特性に応じて、テスト方法やテスト環境が異なる。

ⓑ　テスト設計の手順



ⓒ　テスト対象プログラムの特性の把握

オンラインリアルタイム処理、バッチ処理など、プログラムの特性によって、テスト方法、テスト環境が異なる。オンラインリアルタイム処理では通信回線関係のハードウェアやソフトウェアの準備が必要であり、大規模システムではテストシミュレーションを含むテスト環境の整備が必要になる。テストに必要なハードウェア構成を検討するために、被テストプログラム、同時使用の共通プログラムを明確にし、テストに使用するライブラリの大きさを見積もる必要がある。

ⓓ　テスト設計の検討内容および留意点

㋐　テストケースの設計

テストの目的に合わせて、必要なテストケースを洗い出す。テストケースには、有効なテスト条件、システムに求められている機能、その機能の範囲外、エラーケースも含める。データの有効範囲と無効範囲の境界に関するエラーが多いので、境界に関するテストケースを必ず含める。

㋑　テスト方法の設計

テストケースの内容を吟味し、それぞれのテストケースを単独でテストするか、組み合わせてテストするかを決定する。テスト順序を決定する。

㋒　テストツールの選定

テストケースを効率よくテストするために必要なテストツールを選定する。テストドライバ、テストデータジェネレータ、スタブ、デバッガ等を選定する。

㋓　テスト手順の設計

テストケースを吟味して、テスト実行の手順を決める。テストの手順によって、効率が大きく異なる。結合テスト、システムテスト、運用テストの場合には熟慮が必要である。

㋔　テストデータの設計

テストの目的を明瞭にし、最も効率よく実行できるテストデータを設計する。少ないデータで効率よくテストするには、テストデータジェネレータで作成する方法、テストチームが作成する方法、ユーザが作成する方法のいずれかの方法を選択することが望ましい。負荷テストの場合は現在使用中のファイルから抽出する方法を採用することが多い。テストの目的に合わせたテストデータの作り方を選択し、テーストケースの選定は、モジュール設計書に基づいて，すべてのロジックパスを一度は通るようなテストケースによって検証を行う。

㋕　テスト結果の想定

テストに使用するファイルやデータベース、出力データについて、テスト前の内容とテスト後の内容を明確にする。

㊐　デバッグ方法の設計

デバッグ方法を検討し、効率よくバグを見つけることができるようにする。デバッグ方法もテストの種類に合わせて検討する。

㊑　テストデータの作成方法

❶　テストデータジェネレータで作成する。
❷　テストチームが作成する。
❸　ユーザが作成する。
❹　現在使用中のファイルから抽出する。
❺　現在使用中の帳票から抽出する。

ⓔ　テスト環境の検討内容および留意点

㋐　ハードウェア

テストの種類に対応させて、テストに必要なハードウェア構成を明確にする。必要な時期を明確にし、前もって準備する。

㋑　ソフトウェア

基本ソフトウェアはハードウェアと同様に、必要な時期を明確にして、前もって準備する。テスト支援ツールは必要なテスト支援ツールを明確にし、導入する。モジュールやプログラム単位のテストツールは、共通に使用できるツールの他に、スタブなどのように、モジュールやプログラム単位のテストツールが必要になる。

㋒　テスト体制

テストチームに対する教育、訓練を考慮して早めに手配する。ユーザの参画は、テストケースの設計、テストデータの作成、テストの実施、テストの結果の検証において適宜必要となる。システムテストの場合、ユーザの参画が重要である。テスト終了の承認手続きと承認者を明確にする。

㋓　消耗品の準備

テストに必要な用紙や事務用品等を準備する。

④　テスト支援ツール

ⓐ　テストデータジェネレータ

被テストプログラムが使う入力ファイルやデータベースをテストケースに合わせて作るときに使われる。作成した内容をリストする機能を持っている。プログラムテスト用のデータを簡

単なパラメータを与えることで自動的に生成する。正常なケースのデータ、異常なケースのデータ、エラーデータが生成できる。オンラインシステムで、対話型で作成できるものもある。

テストデータの作成方法は対話型でデータを１件ずつ作成する。ある条件を与えて（初期値と増分などの条件）、データの入力なしに、規則に従ってデータを大量に作る方法がある。作成したデータをファイルに格納するため、ファイルの定義や編成法、フィールドの定義、属性の定義を行っておく。データチェック機能はテストデータ作成時にデータの属性をチェックする。データの再作成は作成済みのデータに条件を与えることによって、目的にあったデータを再作成する。必要項目を指定することで、ファイルのレイアウトの変更も可能である。

ⓑ　カバレージモニタ

テスト対象プログラムのうち、既テスト分と未テスト分を識別するツールである。あるテストデータがプログラムのどの経路を通ったかを調べながら、プログラム全体の経路のうち約何％をカバーしたかを求めることができる。プログラムの経路は選択や繰返しなどの分岐のある文がでてくることによって増えていく。すべての経路をテストすることは不可能なので、カバレージを見ながら判断する。

ⓒ　ドライバとスタブ

㋐　ドライバ

ドライバはボトムアップテストで下位モジュールのテスト時に上位モジュールの代わりに用いるテストプログラムである。

㋑　スタブ

スタブはトップダウンテストで上位モジュールのテスト時に下位モジュールの代わりに用いるテストプログラムである。

ⓓ　デバッガ

デバッガはプログラムのバグを見つけるために使用するテストツールである。プログラム実行の中断、パフォーマンスの分析、実行の継続、コード部のスキップ、エラーの訂正、変数の表示及びセット、期待される入力の設定、出力の表示等の機能がある。

ⓔ　インスペクタ

インスペクタはプログラムを実行してエラー検出やデータ構造の内容を確認するためのデバッギングツールで、プログラムを途中で中断し、トレース対象データの閲覧や更新を対話形式で処理する。

ⓕ　トレーサ

トレーサはプログラムの実行時に制御の流れを追跡するときや実行過程を時系列的に１ステ

ップずつモニタリングするときに使用する。あるデータを処理するときのプログラムの動きを命令単位で調べられる。特定区間の命令を実行する毎に、その所在や命令自身、実行直後のレジスタなどの内容を書き出す。誤りの箇所が特定できない時に有効である。区間の指定を的確にしないと、出力量が膨大になり、処理時間がかかる。ワンステップずつ追跡調査できる。

ⓖ　ダンプツール

ダンプツールはダンプ命令を組み込んで、特定の領域をリストするツールである。特定の命令が実行される前後にダンプ命令を組み込んでプログラムの実行状況を把握するときに使用する。ダンプ命令には、そのプログラムを使っている全ての領域をダンプするものと、指定した部分だけダンプするものがあり、プログラムを実行しながら出力する動的ダンプとプログラムの実行終了後に出力する静的ダンプがある。

ⓗ　スナップショットダンプ

スナップショットダンプはプログラムにデバッグ命令を組み込んでおき、デバッグ命令を実行する都度、主記憶装置の一部やレジスタの内容を書き出す。指定した条件のときだけ主記憶やレジスタの内容などの追跡データを出力するダンプである。

動的ダンプのときに使われ、プログラムにエラーがあり原因不明で分からないとき、プログラムの特定の要所にダンプの条件を設定しておき、そのプログラムの記憶装置よりメモリの一部または全部の内容を出力して、エラーを調べることをいう。

プログラムの誤りの箇所がほぼ見当がついていて、どのような状態で異常が発生しているかを知るため、誤りと思われる命令群の処理前のデータの状態と処理後のデータの状態を比較するときに使用するデバッグツールである。

ⓘ　クロスリファレンス

クロスリファレンスはプログラムで使用している変数や関数の名前を相互参照できるデバッギングツールである。

ⓙ　プリティプリンタ

プリティプリンタは字下げや予約語のフォント変更などプログラムの整形機能をもつデバッギングツールである。

ⓚ　プルーフリスト

プルーフリストは入力したデータが誤りなく入力されているかをチェックするために、入力したデータをそのまま出力して正しく校正する目的で使用するリストである。

⑤　ユニットテスト(単体テスト)

ⓐ　ユニットテストとは

㋐　ユニットテスト

ユニットテストはプログラムの最小単位であるモジュールの品質をテストすることであり、その目的は結合テスト前にモジュール内のエラーを発見することである。テストは機能テストと構造テストの２つの観点から行う。モジュールはプログラムを構成する要素であるから、単体では動作しない。ドライバとスタブというテスト支援ツールを使用してテストを行う。

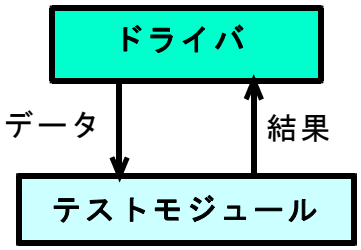
㋑　機能テスト

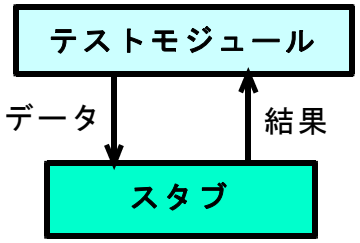
機能テストはモジュールの機能仕様をもとに、すべての入力条件や出力条件、エラー処理など、モジュールの機能が満足されているかどうかを検証する。機能テストは外部に見える機能の検証である。

㋒　構造テスト

構造テストはモジュールの詳細仕様や原始プログラムをもとに、モジュールの論理が正しいかどうかの検証を行う。プログラム内部の論理の検証である。

ⓑ　ドライバとスタブ





㋐　ドライバ

複数のソフトウェア部品（モジュール）の結合テストを行なう際に、呼び出し側のモジュールが未完成であったりテストのために実行するのが面倒だったりする場合に、その代用となるテスト用の呼び出しプログラムをテストドライバ、あるいは単にドライバという

ドライバはテストモジュールの上位モジュールの機能をシミュレートする。テストモジュールから見て主プログラムの役割を果たす。結合テストのボトムアップテストに利用される。

㋑　スタブ

モジュールの動作をテストする際、呼び出し先の下位モジュールの代わりをする空のモジュールのことをスタブという。下位モジュールが未完成なうちに上位モジュールのテストをしたい場合に用意される。スタブはテストモジュールの下位モジュールの機能をシミュレー

トするもので、テストモジュールから見て副プログラムの役割を果たす。結合テストのトップダウンテストに利用される。

スタブは下位モジュールと同じ名称や引数、返り値の型などを持ち、上位モジュールから同じコードで呼び出される。内部は空で何も処理を行わないが、本物と同じような値を返さなければならない場合には、想定される返り値の一つをあらかじめ算出しておき、これを定数として機械的に返却するといった処理を行う。

⑥　ブラックボックステスト

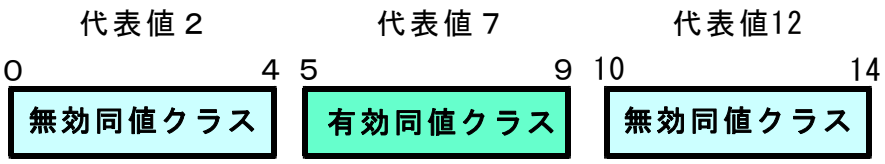
ⓐ　ブラックボックステストとは

ブラックボックステストはプログラムの外部仕様をもとにテストケースを設計するための技法で、プログラムの詳細なアルゴリズムの仕様は参照しないで、プログラムの機能仕様やインタフェース仕様だけを用いて設計する。テストケースの設計はエラーを検出する可能性の高いデータを効果的に組み合わせて、効率的にテストが行えるようにすることである。稼働後のシステムの品質はテスト時に使用したテストケースの良否に左右される。プログラムの設計内容から、プログラムの機能とデータの関係を考慮して、テストデータを作成し、プログラムのテストを行う。技法として、同値分割、限界値分析、因果グラフ、実験計画法などがある。

ⓑ　因果グラフ

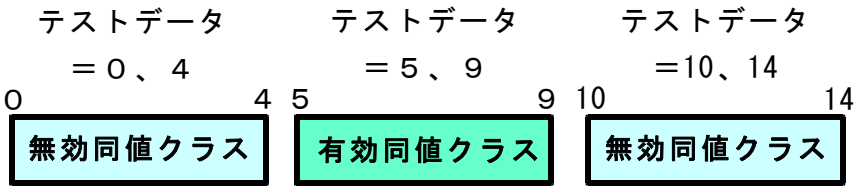
因果グラフは、テスト対象の入力が明確にクラス分けできないようなときに、入力・出力、原因・結果の関係をグラフで表現し、デシジョンテーブルに展開して、テスト項目を設計する。基本要素として、同値、否定、和、積、必要とする、排他的、包含するなどの論理関係を利用してグラフ展開する。

ⓒ　同値分割



同値分割は、テスト対象の入力データの取り得る値の範囲の中から、同じ意味を持つ範囲を１つのクラスとして、いくつかのクラスに分割する。分割したクラスの中から、各クラスを代表する値をテストデータとして選択する。クラス分割する場合、入力データの正しいものおよび誤っているものについてもいくつかのクラスに分割する。正しい範囲のテストデータを有効同値クラス、誤ったデータの範囲を無効同値クラスという。

ⓓ　限界値分析



限界値分析は、入力データ、出力データを同値クラスに分割し、それぞれのクラスの境界条件をテストの対象データになるように値を選ぶ方法である。同じクラス内の最大値または最小値、あるいは両方の値を選ぶ方法である。テストデータは入力条件だけでなく出力も意識して作成する必要がある。

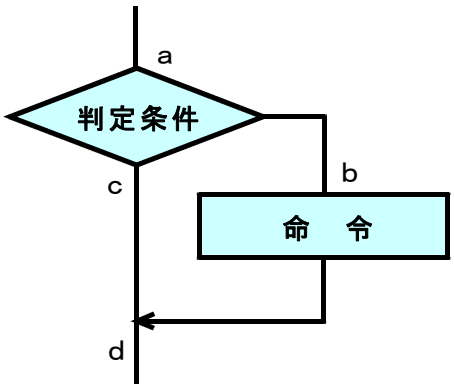
⑦　ホワイトボックステスト

ⓐ　ホワイトボックステストとは

ホワイトボックステストはプログラムの制御の流れに着目し、プログラムのステップの重要な部分を通るようなテストデータを作成し、テストする方法である。プログラムの内部仕様をもとにして、テストケースを設計する技法で、プログラムの内部構造や論理を詳細に調べるため、プログラマの立場から見た詳細な機能テストは行えるが、仕様にはあるがプログラムに実現されていない機能のエラーを発見できない問題がある。

規模の大きいプログラムでは、代表的な正常処理の経路と異常処理や例外処理の経路を中心にテストケースを設計する。すべてのステップを網羅するテストデータは膨大になるため、命令網羅、判定条件網羅、条件網羅、複数条件網羅などの簡略化した方法を利用してテストケースを設計する。

ⓑ　命令網羅



命令網羅は条件式の真偽に関係なく、すべての命令を少なくとも１回は実行するようにテストケースを設計する。繰り返しの対象のブロック内の命令も最低１回は実行させる。流れ図で

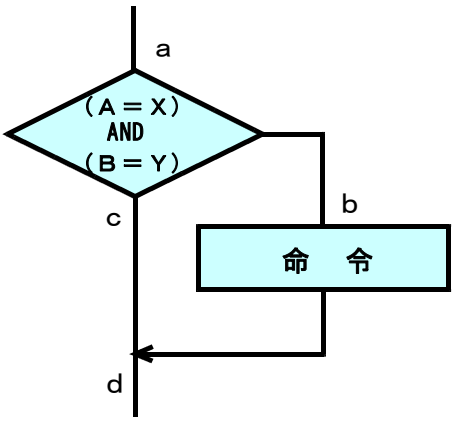
ａ→ｂ→ｄの経路のテストを実行すれば、命令を網羅したことになる。

ⓒ　判定条件網羅

判定条件網羅は条件式の真偽ではなく、判定結果の真偽または判定結果の種類の数の経路を網羅し、かつすべての命令を少なくとも１回は実行するようにテストケースを設計する。流れ図で、ａ→ｂ→ｄ、ａ→ｃ→ｄの２つの経路のテストを実行すれば、命令を網羅したことになる。

ⓓ　条件網羅

条件網羅は、判定結果ではなくすべての条件式において真偽を判定し、かつすべての命令を１回は実行するようにテストケースを設計する。流れ図で、判定条件(Ａ＝Ｘ)AND(Ｂ＝Ｙ)が真の場合と偽の場合の２通りの組み合わせをテストする。真の場合は、(Ａ＝Ｘ)AND(Ｂ＝Ｙ)が成り立つ場合であるが、偽の場合は、(Ａ≠Ｘ)AND(Ｂ＝Ｙ)、(Ａ＝Ｘ)AND(Ｂ≠Ｙ)、(Ａ≠Ｘ)AND(Ｂ≠Ｙ)の３ケースあるが、このうち条件式が偽となる１ケースについて実行すればよいことになる。



ⓔ　複数条件網羅

複数条件網羅は、それぞれの判定における条件付きの可能なすべての組み合わせ、かつすべての命令を少なくとも１回は実行するようにテストケースを設計する。流れ図において、判定条件は、真の場合は、(Ａ＝Ｘ)AND(Ｂ＝Ｙ)が成り立ち、偽の場合は、(Ａ≠Ｘ)AND(Ｂ＝Ｙ)、(Ａ＝Ｘ)AND(Ｂ≠Ｙ)、(Ａ≠Ｘ)AND(Ｂ≠Ｙ)の３ケースが成り立つ。これらの４ケースすべてについてテストする。

⑧　結合テスト

ⓐ　結合テストとは

結合テストはモジュールを結合したテストで、モジュール間のインタフェースの検証とプロ

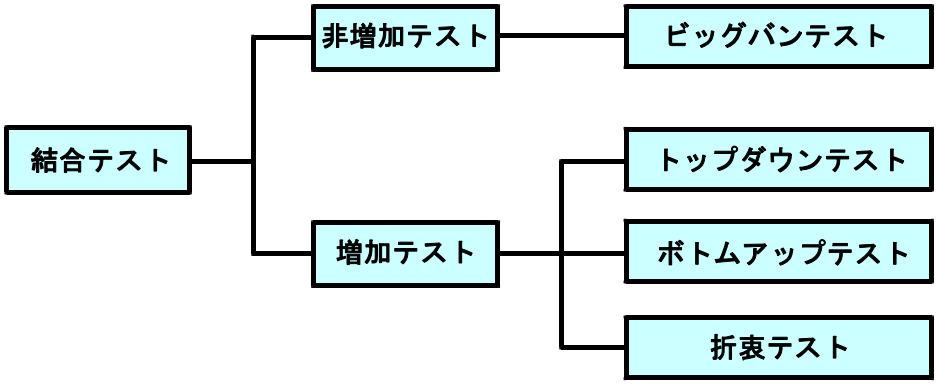
グラムの入出力を含むテストである。プログラムが外部仕様で定められた機能どおりに、実現されているかを検証する。信頼性や操作性、保守性などの検証も行う。

ⓑ　結合テストの手法

結合テストの手法には、すべてのモジュールを一斉に結合して行う非増加テストのビッグバンテストとテストの進行につれて結合するモジュールを逐次増加させながら行う増加テストの方法がある。

増加テストには、上位のモジュールからスタートして、逐次下位モジュールを結合しながらテストを進めるトップダウンテストと、逆に、下位モジュールからスタートして、逐次上位モジュールを結合しながらテストを進めるボトムアップテストがある。また、トップダウンテストとボトムアップテストの両方の手法を組み合わせてテストする折衷テストがある。

どの方式を用いてテストを行うかは、開発しているソフトウェアの特徴や要求される信頼性、開発の進捗状況など各種の条件を考慮して決定する。



⑨　ビッグバンテスト

ⓐ　ビッグバンテストとは

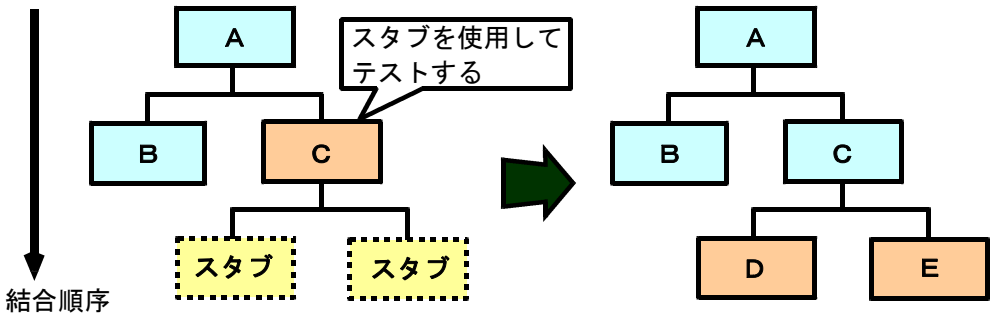
ビッグバンテストはモジュール単体毎にドライバまたはスタブを用意し、すべてのモジュールテストを行い、最後に、各モジュールを結合して一斉にプログラムテストを行う方法である。ドライバやスタブの作成量が多いため、経済性の面で増加テストより劣る。モジュールテストが十分に行われるため、信頼性は高い。

ⓑ　ビッグバンテストの特徴

㋐　結合テスト時に、ドライバやスタブが不要である。
㋑　骨組みになる部分も含めて、すべてのモジュールが、最初からテストされる。
㋒　特定の経路をテストすることが難しい。
㋓　エラーが発生したときのデバッグが容易でない。
㋔　単体テストが終了していないモジュールが１個でもあると、結合テストを開始できない。

⑩　トップダウンテスト

ⓐ　トップダウンテストとは



トップダウンテストは最上位のモジュールからモジュールテストを行う方式で、順次、下位モジュールを結合しながらモジュールテストを繰り返す。上位モジュールからテストを行うため、下位モジュールが未完の場合、スタブが必要である。スタブは、上位モジュールから制御をもらい、引数を設定してから制御を戻す簡単な機能を持つ。トップダウンテストはモジュール間インタフェースを仮定する必要が無く、引数の受渡をテストすることができる。最上位モジュールは最初からテストが繰り返されており、バグの発見も効率よく行え、プログラム全体の制御を司る重要な部分がきちんとテストできる。

ⓑ　トップダウンテストの特徴

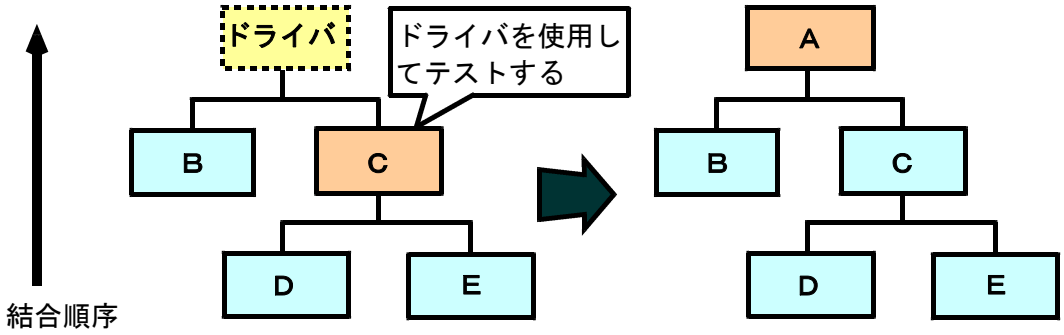
㋐　最初は並行作業が困難であり、テストするまで時間が空いてしまう可能性がある。
㋑　上位のインタフェースが早い時期にテストできる。
㋒　重要度の高い上位モジュールが繰返し実行されるので、信頼性が高い。
㋓　モジュール単体の機能や、個々の論理を十分にテストできないことがある。
㋔　一般に、新規開発システムに適用すると効果がある。
㋕　トップダウンでモジュールの集積を進めるため、並行した開発がし難い。
㋖　全体としては工数が多くなる。

⑪　ボトムアップテスト

ⓐ　ボトムアップテストとは

ボトムアップテストはプログラム構造の最下位レベルのモジュールを最初にテストし、次に、それらの上位レベルのモジュールを結合してテストする。これを繰り返しながら最後に最上位のモジュールを結合してテストを行う。上位モジュールが未完の場合、テストモジュールの上位モジュールの機能をシミュレートするドライバが必要である。ドライバは、下位モジュール

に制御と引数を渡すとともに、下位モジュールから戻ってきた引数をもとに、対応する操作を実行する。上位モジュールの役割を代行する機能を持つ。インタフェーステストを何回も行う場合、繰り返し下位モジュールを呼び出せる制御構造を持つ必要があり、ドライバの作成が煩雑になる。



ⓑ　ボトムアップテストの特徴

㋐　開発の初期の段階から並行作業が可能である。
㋑　モジュール単体の機能や論理が十分にテストできる。
㋒　テストの最終段階でインタフェース上の問題が発生しやすい。
㋓　全体の機能を把握するのに時間がかかる。
㋔　既に稼働しているシステムを修正して、システムを開発する場合に有効である。
㋕　問題は、モジュール間のインタフェースについてすべてを結合しない限り、全体の整合性をテストできない点にある。
㋖　最上位モジュールが最後に結合されるため、十分にテストできない。

⑫　折衷テスト

ⓐ　折衷テストとは

折衷テストはトップダウンテストとボトムアップテストを組み合わせてテストを行う方法である。最上位モジュールはトップダウンテスト、最下位に近いモジュールはボトムアップテストを用いる。トップダウンテストのモジュール間のインタフェース活用の利点とボトムアップテストの並行テストの実施の利点を組み合わせて行える。

ⓑ　折衷テストテストの特徴

㋐　並行してテストできる程度が高い。
㋑　骨組みになる部分のテストが早くからできる。
㋒　テストの計画及び管理がやりやすい。
㋓　特定の経路のテストが容易である。

㋓　ドライバ、スタブの両方を用意する必要がある。

㋔　最上位のモジュールと最下位のモジュールの両方からテストを進めることが可能で、中間に位置するモジュールが最後にテストされる。

㋕　全ての下位モジュールをスタブとしてあらかじめ作成しておくことができる。

⑬　システムテスト

ⓐ　システムテストとは

システムテストはソフトウェアやシステムが要求された仕様に合致しているかどうかを検証するために、複数の機能グループを組み合わせて行うテストである。

ⓑ　システムテストの内容

㋐　プログラム間やサブシステム間の結合テスト

㋑　サブシステム・システムの入出力を含むテスト

㋒　例外データを与えたときに正しく例外処理を行うことを確認する例外事項のテスト

㋓　障害テスト

㋔　性能テスト(スループット、レスポンスタイム、ターンアラウンドタイム)、負荷テスト

㋕　機能テスト

ⓒ　主要システムテストの検討内容

㋐　性能テスト

スループット、レスポンスタイム、ターンアラウンドタイムなどを評価する。スループットは単位時間当たりの処理する仕事の量で、同時に稼働している端末数やマルチプログラミングの性能によって変化するため、各種条件での検証が必要になる。レスポンスタイムも同時に稼働している端末装置の台数によって異なるため、各種条件での検証が必要になる。

㋑　負荷テスト

短時間に重い負荷をかけるテストである。同時に実行するプログラム数や同時に稼働する端末台数を増加させて、レスポンスタイムなどをテストする。システムが負荷に耐えられるかどうかをテストする。

㋒　機能テスト

ユーザの要求仕様を満足しているかどうかをテストする。ユーザの要求仕様の機能が詳細に記述されていない場合には、設計書の中から要求仕様を取り出す作業が必要になる。

⑭　運用テスト

ⓐ　運用テストとは

運用テストはユーザ部門の運用グループが実際の運用と同一の条件を作り出し、機能面および操作面の双方のテストを行う。承認テスト、導入テスト、フィールドテストなどがある。ある程度の期間を費やし、担当者がシステムの操作に習熟する研修を兼ねて行われることもある。

運用テストでは、本番移行基準の確認、移行テスト、保守性テスト、運用効率や業務効率の測定、ユーザビリティテストなどが行われる。移行テストでは、安全性・効率性の観点で、既存システムから新システムへの切り替え手順、切り替えに伴う問題点を確認する。

仕様書通りの論理が実装されていることを確認するだけではなく、操作に対する応答時間や単位時間当たりの処理性能を計測したり、高い負荷をかけたときの反応や耐久性を見たり、入力ミスや誤操作、ハードウェア障害などを故意に発生させてエラー処理や復旧などの手順を確認したりする場合もある。

ⓑ　承認テスト

開発部門がユーザ部門の承認を得るテストである。ユーザ部門がデータを用意し、要求仕様どおりの結果が得られるかどうかをユーザ部門に承認してもらうために行う。ユーザ部門が管理、実行する。

ⓒ　導入テスト

システムが実際の動作環境で、誤りなく稼働するかどうかを確認するテストで、実際の業務に即した利用の仕方をしてみて問題なく動作するかを試すテストである。承認テスト、検収テストを兼ねる場合もある。

ⓓ　フィールドテスト

フィールドテストは特定の利用者に実際にシステムを使ってもらう実地テストである。開発した製品を、開発部門以外の部門で、製品の仕様書と照らし合わせて評価・確認するテストである。特定の利用者に、実際の仕事に使ってもらう実地テストである。

⑮　その他のテスト

ⓐ　移行テスト

移行テストは、現行システムから新システムへ、ハードウェアやソフトウェア、通信回線、各種ファイルを円滑に移し変えるために、移行方法や移行手順、移行体制、移行日程計画、移行タイムチャート、移行対象データ項目、データの移行方法、ＯＳやミドルウェアの入替などの問題点を検討するために行うテストである。

ⓑ　コードインスペクション

コードインスペクションは机上でプログラムを詳細に読んで検査することをいう。エラーをタイプ別に分類、集計し、次に発生する同種のエラーの発見を容易にするために利用する。ＯＳのように、高い信頼性を要求されるソフトウェアの開発に利用する。

ⓒ　リグレッションテスト（退行テスト）

レグレッションテストはシステムの保守段階に行うテストで、プログラムの変更により、既存の正しい範囲に新しい誤りが発生しないかどうかを検証する。コンピュータプログラムに手を加えたことによる影響を確認するテスト操作である。

プログラムが大規模化で複雑化すると、何も関係がないかのように見えるプログラムが相互に関係しあっているのを見落とす場合も少なくない。ある箇所を改善しようとして加えた修正が、思いもよらない部分に影響してバグを呼び起こしてしまう。

ⓓ　ペネトレーションテスト

ペネトレーションテストは、コンピュータやネットワークのセキュリティ上の弱点を発見するテスト手法の一つで、システムを実際に攻撃して侵入を試みる手法である。

ネットワーク接続された情報システムが外部からの攻撃に対して安全かどうか、実際に攻撃手法を試しながら安全性の検証を行う。不正に侵入できるかどうかだけでなく、ＤｏＳ攻撃にどれくらい耐えられるかを調べたり、侵入された際にそこを踏み台にして他のネットワークを攻撃できるかどうかなどを調べる場合もある。

例題演習

プログラムのテストの目的として，最も重要なものはどれか。

ア　バグがないことを示すこと　　イ　バグの原因を究明すること
ウ　バグを修正すること　　エ　バグを見つけること

解答解説

テストの目的に関する問題である。

アのバグがないことを示すことは、テストの結果から諸条件を考えて論理的に展開することは可能であるが、現状ではバグが０であると断定することは難しい。テストを実施することの目的とは異なる。

イのバグの原因追及は、バッグの再発生を防止するために考える内容であり、テスト結果に基づいて次のステップで考える内容である。

ウのバグの修正は、テストの結果に基づいて、改善の対策を考える段階の問題であり、テストの直接の目的ではない。

エのバグを見つけることがテストの最も重要な目的である。求める答えはエとなる。

例題演習

プログラムモジュールの単体テストに関して、正しい記述はどれか。

ア　トップダウンテストでは、テスト対象のプログラムモジュールが呼び出す下位モジュールの代わりをするスタブが必要である。

イ　入力条件のテストでは、プログラム設計で規定された最大値・最小値のケースが重要であり、明らかに誤った条件の入力ケースを実施する必要がない。

ウ　プログラムモジュール１本ごとの論理上の正しさを証明するものであるから、コンパイルでエラーが発生しなければ単体テスト完了とする。

エ　プログラムモジュールのコーディングが全て完了していなくても、単体テストを開始することができる。

解答解説

単体テストに関する問題である。

アのトップダウンテストでは下位モジュールの代わりをするスタブが必要であり、アの記述は正しい。求める答えはアとなる。

イの入力条件に関するテストはモジュールの機能仕様に基づいて、すべての入力条件、出力条件、エラー処理等、モジュールの持つ機能が満足しているかどうかを検証する。従って、誤った条件の入力ケースを実施する必要がないという記述は誤りである。

ウは、プログラム内部の論理の検証が必要である。コンパイルでエラーが発生しないだけでは論理上十分であるとはいえない。

エは、コーディングが完了していないとコンパイルできないため単体テストができない。

例題演習

入力データと出力結果の関係に注目してテストデータを作成し、プログラムの機能をテストする手法はどれか。

ア　トップダウンテスト　　イ　ブラックボックステスト

ウ　ボトムアップテスト　　エ　ホワイトボックステスト

解答解説

プログラムの機能をテストするブラックボックステストに関する問題である。

アのトップダウンテストは、上位のモジュールから下位のモジュールへと順次結合して行う結合テストである。

イのブラックボックステストは、モジュールをブラックボックスと見なし、機能仕様書に基づき作成したテストデータでテストを行う手法である。入力データと出力結果の関係に注目して、プログラムの機能をテストするものである。求める答えはイである。

ウのボトムアップテストは、下位のモジュールから上位のモジュールへと順次結合して行う結合テストである。

エのホワイトボックステストは、モジュールの制御構造を詳細に検討するテストである。

例題演習

ソフトウェア開発におけるテスト技法のうち，ブラックボックステストに関する記述として，適切なものはどれか。

ア　原始プログラムを解析し，プログラムの制御の流れと変数などのデータの流れをテストするものであり，主にプログラム開発者以外の第三者が実施する。
イ　ブログラムが設計者の意図した機能を実現しているかどうかのテストであり，主にプログラム開発者以外の第三者が実施する。
ウ　プログラムのすべての命令が最低１回は実行されることを目的とするテストであり，主にプログラム開発者自身が実施する。
エ　プログラムの内部構造や論理が記述された内部仕様書に基づくテストであり，主にプログラム開発者自身が実施する。

解答解説

ブラックボックステストに関する問題である。

ブラックボックステストは、プログラムの外部仕様をもとにテストケースを設計するための技法で、プログラムの詳細なアルゴリズムの仕様は参照しないで、プログラムの機能仕様やインタフェース仕様だけを用いて設計する。プログラムの設計内容から、プログラムの機能とデータの関係を考慮して、テストデータを作成し、プログラムのテストを行う。技法として、同値分割、限界値分析、因果グラフ、実験計画法などがある。

アは机上デバック、イはブラックボックステスト、ウ、エはホワイトボックステストの考え方である。求める答えはイとなる。

例題演習

プログラムテストにおける限界値分析で設定するテストデータとして、適切なものはどれか。ここで、“Ａの直前の値”とは“Ａより小さくてＡに近い値”を指し、“Ａの直後の値”とは“Ａより大きくてＡに近い値”を指す。

ア　最小値、最小値の直後の値、最大値の直前の値、最大値
イ　最小値、最大値
ウ　最小値の直前の値、最小値、最大値、最大値の直後の値
エ　最小値の直前の値、最小値の直後の値、最大値の直前の値、最大値の直後の値

解答解説

限界値分析におけるテストデータに関する問題である。

限界値分析は、ブラックボックステスト法の一手法であり、入力データと出力データを同値クラスに分割し、それぞれのクラスの端がテストの対象になるように値を選ぶ方法である。通常は有効同値クラスの最大・最小とそれぞれを一つ超えた値を用いる。従って、最小値の直前の値、最小値、最大値、最大値の直後の値となる。求める答えはウとなる。

例題演習

表は，あるプログラムの入力データを，有効同値クラスと無効同値クラスに分けたものである。同値分割法によってテストケースを設計する場合，最小限のテストデータの組合せとして，適切なものはどれか。

ア　－２，０，１，５，６，８
イ　０，１，５，６
ウ　－１，３，６
エ　１，５

同値クラス	データ
無効同値クラス	−2, −1, 0
有効同値クラス	1, 2, 3, 4, 5
無効同値クラス	6, 7, 8

解答解説

同値分割法に関する問題である。

同値分割法は、テスト対象の入力データの取り得る値の範囲の中から、同じ意味を持つ範囲を１つのクラスとして、いくつかのクラスに分割する。分割したクラスの中から、各クラスを代表する値をテストデータとして選択する。

無効同値クラス－２～０、６～８の中からそれぞれ１つずつ選択し、有効同値クラス１～５の中から１つ選択しテストデータとする。－１、３、６のテストデータの組み合わせが適切である。求める答えがウとなる。

例題演習

ホワイトボックステストのテストデータの作成方法に関する記述として、適切なものはどれか。

ア　同値分割の技法を使用してテストデータを作成する。
イ　プログラムの外部仕様に基づいてテストデータを作成する。
ウ　プログラムの内部構造に基づいてテストデータを作成する。
エ　プログラムの入力と出力の関係からテストデータを作成する。

解答解説

ホワイトボックステストに関する問題である。

ホワイトボックステストは、プログラムの制御の流れに着目し、プログラムのステップの重要な部分を通るようなテストデータを作成し、テストする方法である。プログラムの内部構造や論理を詳細に調べるため、プログラマの立場から見た詳細な機能テストは行えるが、仕様にはあるがプログラムに実現されていない機能のエラーを発見できない問題がある。規模の大きいプログラムでは、代表的な正常処理の経路と異常処理や例外処理の経路を中心にテストケースを設計する。すべてのステップを網羅するテストデータは膨大になるため、命令網羅、判定条件網羅、条件網羅、複数条件網羅などの簡略化した方法を利用してテストケースを設計する。

ア、イ、エはブラックボックステストに関する内容であり、ウがホワイトボックステストに関する内容である。求める答えはウとなる。

例題演習

ホワイトボックス法に属するテストケースの作成方法はどれか。

ア　原因－結果グラフ　　イ　限界値分析

ウ　条件網羅　　エ　同値分割

解答解説

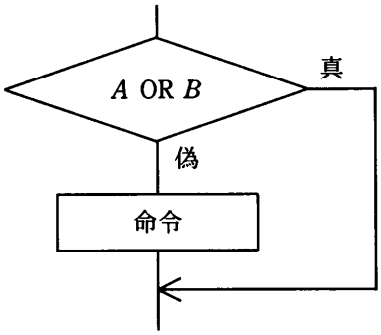
ホワイトボックステストのテストケースに関する問題である。

ホワイトボックステストは、ログラムのテストデータ選択方法の１つで、プログラムのモジュールの実行経路を詳細に確認するためにテストデータを選択する。すべての命令をテストすることは難しく、要求されている機能の確認も見出しにくい問題がある。テストデータの作成方法に、命令網羅、判定条件網羅、条件網羅、複数条件網羅等がある。

ア、イ、エはブラックボックス法のテストケース作成方法で、ウの条件網羅がホワイトボックス法のテストケース作成方法である。求める答えはウとなる。

例題演習

図の論理を判定条件網羅（分岐網羅）でテストするときのテストケースとして，適切なものはどれか。



ア

A	B
偽	真

イ

A	B
偽	真
真	偽

ウ

A	B
偽	偽
真	真

エ

A	B
偽	真
真	偽
真	真

解答解説

ホワイトボックステストに関する問題である。

判定条件網羅は、プログラムの全ての判定条件で、真と偽を少なくとも１回以上実行するようにテストケースを設計する。

アの場合は真の場合のテストのみである。

イの場合は、Ａが偽Ｂが真の場合も、Ｂが偽Ａが真の場合も共にＯＲは真であるから、真の場合のテストのみである。

ウの場合は、Ａが偽Ｂが偽の場合はＯＲは偽、Ａが真Ｂが真の場合はＯＲは真であるから、真と偽を少なくとも１回実行していることになる。真、偽を少なくても１回行うテストの判定条件網羅はウとなる。求める答えはウとなる。

エの場合は３回のテストは全て真の場合である。

例題演習

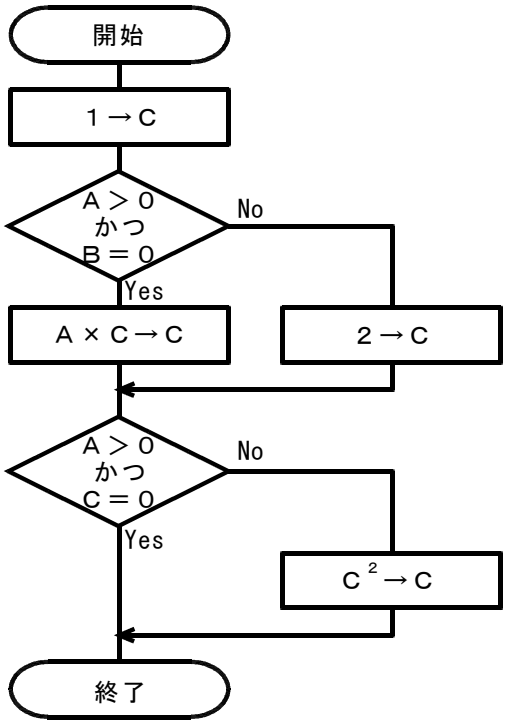
次の流れ図において、判定条件網羅（分岐網羅）を満たす最少のテストケースはどれか。

ア　（１）Ａ＝０，Ｂ＝０
　　（２）Ａ＝１，Ｂ＝１

イ　（１）Ａ＝１，Ｂ＝０
　　（２）Ａ＝１，Ｂ＝１

ウ　（１）Ａ＝０，Ｂ＝０
　　（２）Ａ＝１，Ｂ＝１
　　（３）Ａ＝１，Ｂ＝０

エ　（１）Ａ＝０，Ｂ＝０
　　（２）Ａ＝０，Ｂ＝１
　　（３）Ａ＝１，Ｂ＝０
　　（４）Ａ＝１，Ｂ＝１



解答解説

ホワイトボックステストの判定条件網羅に関する問題である。

判定条件網羅は、プログラムのすべての判定条件で真と偽を少なくとも１回以上実行するようにテストケースを設計する。

この流れ図では次の条件の内容に分けることができる。

①　最初の判定条件の真の場合はＡ＝１かつＢ＝０

②　最初の判定条件の偽の場合はＡ＝１かつＢ＝１、Ａ＝０かつＢ＝１、Ａ＝０かつＢ＝０のいずれか。

③　二つ目の判定条件の真の場合はＡ＝１かつＣ＝０

④　二つ目の判定条件の偽の場合はＡ＝１かつＣ＝１、Ａ＝０かつＣ＝１、Ａ＝０かつＣ＝０のいずれか。

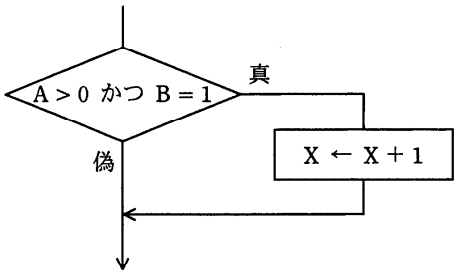
この流れ図ではＣ＝１であるから、２番目の判定条件はすべて偽となる。従って、判定条件網羅では最初の判定条件の真の場合と偽の場合の内の１つを実行すればテスト条件は十分になる。

肯定のＡ＞０かつＢ＝０はＡ＝１かつＢ＝０で１ケース、否定はＡ＝１かつＢ＝１で否定の条件判定を行う。求める答えはイとなる。

例題演習

図の構造をもつプログラムに対して，ホワイトボックステストのテストケースを設計するとき，少なくとも実施しなければならないテストケース数が最大になるテスト技法はどれか。

ア　条件網羅
イ　判定条件網羅
ウ　複数条件網羅
エ　命令網羅



解答解説

ホワイトボックステストのテストケースに関する問題である。

アの条件網羅は、処理の分岐条件で条件式に着目し、真偽について少なくとも１回は実行するようにテストケースを作成する。判定結果よりも条件式に着目する。最低２回テストする。

イの判定条件網羅は、分岐条件の判定結果である真偽の両方の判定を少なくとも１回実行するようにテストケースを設計する。最低２回テストする。

ウの複数条件網羅は、分岐条件の判定結果の真偽を通過するあらゆる組合せを網羅するテストケースを作成する。最低４回テストする。

エの命令網羅は、プログラム中のすべての命令を１回は実行するようにテストケースを作成する。最低１回はテストする。

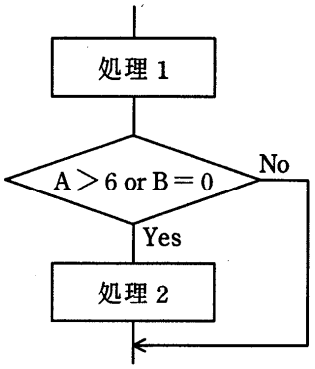
テストケースが最大になるのは複数条件網羅である。求める答えはウとなる。

例題演習

プログラムの流れ図で示される部分に関するテストデータを，判定条件網羅（分岐網羅）によって設定した。このテストデータを複数条件網羅による設定に変更したとき，加えるべきテストデータのうち，適切なものはどれか。ここで，（　）で囲んだ部分は，一組のテストデータを表すものとする。

・判定条件網羅（分岐網羅）によるテストデータ
　（Ａ＝４，Ｂ＝１），（Ａ＝５，Ｂ＝０）

ア　（Ａ＝３，Ｂ＝０），（Ａ＝７，Ｂ＝２）
イ　（Ａ＝３，Ｂ＝２），（Ａ＝８，Ｂ＝０）
ウ　（Ａ＝４，Ｂ＝０），（Ａ＝８，Ｂ＝０）
エ　（Ａ＝７，Ｂ＝０），（Ａ＝８，Ｂ＝２）



解答解説

ホワイトボックステストの条件網羅に関する問題である。

判定条件網羅の場合、（Ａ＝４、Ｂ＝１）はＡ、Ｂ共に偽で、流れ図のＮｏの処理になる。（Ａ＝５、Ｂ＝０）はＡは偽、Ｂは真であり、ＯＲであるから流れ図の処理はＹｅｓになる。

複数条件網羅にする場合、Ａ、Ｂ共に真、Ａは真、Ｂは偽の２つのテストケースを追加する必要がある。

アの場合、（Ａ＝３，Ｂ＝０）はＡは偽、Ｂは真であり、（Ａ＝７，Ｂ＝２）はＡは真、Ｂは偽となる。

イの場合、（Ａ＝３，Ｂ＝２）はＡは偽、Ｂは偽であり、（Ａ＝８，Ｂ＝０）はＡは真、Ｂは真となる。

ウの場合、（Ａ＝４，Ｂ＝０）はＡは偽、Ｂは真であり、（Ａ＝８，Ｂ＝０）はＡは真、Ｂは真となる。

エの場合、（Ａ＝７，Ｂ＝０）はＡは真、Ｂは真であり、（Ａ＝８，Ｂ＝２）はＡは真、Ｂは偽となる。求める答えはエとなる。

例題演習

プログラム中に次の複合判定がある。

条件１　ＯＲ　（条件２　ＡＮＤ　条件３）

判定条件網羅（分岐網羅）に基づいてテストする場合，追加するテスト項目として，適切なものはどれか。

〔終了したテスト項目〕

（１）　条件１が真，条件２が偽，条件３が偽

（２）　条件１が偽，条件２が真，条件３が真

	条件1	条件2	条件3
ア	偽	偽	真
イ	真	偽	真
ウ	真	真	偽
エ	真	真	真

解答解説

判定条件網羅に関する関する問題である。

判定条件網羅は、判定結果が真または偽となるすべての経路を網羅し、かつすべての命令を少なくとも１回は実行するようにテストケースを設計する。

このテスト内容で判定条件網羅の条件を満たすのは、真となるのは次の３つの内の１つが成り立てばよい。

１．条件１または条件２と条件３が同時に真となる。

２．条件１が真となる。

３．条件２と条件３が同時に真となる。

偽となる条件は次のいずれかが成り立てばよいことになる。

４．条件１が偽、条件２または条件３のどちらかが偽になる。

５．条件１、条件２、条件３いずれもが偽になる。

終了したテスト項目では(1)が2の場合、(2)が3の場合で、偽となる条件のテストが行われていない。従って、条件1が偽、条件2が偽、条件3が真のテストが必要である。求める答えはアとなる。

例題演習

プログラムの構造や制御の流れに着目し、プログラム内のすべての経路を網羅するようなテストを行うのはどれか。

ア　トップダウンテスト　　イ　ブラックボックステスト
ウ　ボトムアップテスト　　エ　ホワイトボックステスト

解答解説

ホワイトボックステストに関する問題である。

アのトップダウンテストは、上位のモジュールから下位のモジュールへと順次結合して行う結合テストである。

イのブラックボックステストは、モジュールをブラックボックスと見なし、機能仕様書に基づき作成したテストデータでテストを行う手法である。

ウのボトムアップテストは、下位のモジュールから上位のモジュールへと順次結合して行う結合テストである。

エのホワイトボックステストは、モジュールの制御構造を詳細に検討するテストである。

プログラムの構造や制御の流れに着目し、プログラム内のすべての経路を網羅するテストはホワイトボックステストで、求める答えはエとなる。

例題演習

モジュール単体テストに関する記述として，最も適切なものはどれか。

ア　通常はコーディングを行ったプログラマではなく，専任のテスト要員がテストケースを作成し，実行する。
イ　モジュール間インタフェースは，モジュール単体ではテストできないので，単体テストの対象外となる。
ウ　モジュール設計書は，正しいことが検証済みであるので，テスト結果に問題があるときは，テストケース又はモジュールに誤りがある。
エ　モジュール設計書を見ながら，原則としてすべてのロジックパスを一度は通るようなテストケースによって，検証を行う。

解答解説

モジュールテストに関する問題である。

アのテストケースの作成は、モジュール設計段階に行うため、プログラマが行う。テストの内容によってユーザが参画することがある。テストデータはテストチームやユーザが作成する。

イのモジュール間インタフェースは、スタブまたはドライバを使用して、行うことができる。

ただし、スタブまたはドライバを使用するので、代替モジュールの結果が適切とはいえない。

ウのモジュール設計書が正しくて、テスト結果に問題がある場合、テストケースやモジュールに誤りがなくても、テストデータやテストの方法、テストツールなどに問題があると、テスト結果に問題が発生する。

エの原則としてすべてのロジックパスを一度通るテストケースで検証を行う記述は適切である。求める答えはエとなる。

例題演習

デバッグツールとして用いるトレーサの説明として、適切なものはどれか。

ア　磁気テープファイルや磁気ディスクファイルなどの内容を出力する。
イ　プログラムの実行中にエラーが発生したとき、メモリの内容を出力する。
ウ　プログラムの特定の命令を実行するごとに、指定されたメモリの内容を出力する。
エ　プログラムの命令の実行順序、実行結果などの履歴情報を出力する。

解答解説

トレーサに関する問題である。

トレーサは、プログラムの実行時に制御の流れを追跡するときや実行過程を時系列的に１ステップずつモニタリングするときに使用する。あるデータを処理するときのプログラムの動きを命令単位で調べたり、特定区間の命令を実行する毎に、その所在や命令自身、実行直後のレジスタなどの内容を書き出したりする場合に用いる。誤りの箇所が特定できない時に有効である。区間の指定を的確にしないと、出力量が膨大になり、処理時間がかかる。ワンステップずつ追跡調査できる。

アはダンプツール、イはデバッガ、ウはスナップショットダンプ、エがトレーサである。求める答えはエとなる。

例題演習

プログラムの動作過程を実行順にモニタリングするデバッギングツールはどれか。

ア　インスペクタ　　イ　クロスリファレンス
ウ　トレーサ　　エ　プリティプリンタ

解答解説

デバッグツールに関する問題である。

アのインスベクタは、プログラムを実行し、エラーを検出する動的デバッグツールで、実行を中断しデータ内容の閲覧や更新を対話形式で処理する。

イのクロスリファレンスは、プログラムで用いられている関数、変数や定数に関する名前を相互に参照する静的デバッギングツールである。

ウのトレーサは、プログラムの実行過程を時系列的に１ステップずつモニタリングする動的デバッギングツールで、プログラムをステップごとにエラーを検証する。プログラムの動作過

程を実行順にモニタリングするのはトレーサである。求める答えはウとなる。

エのプリティプリンタは、字下げや予約語のフォント変更など、プログラムの整形機能をもつ静的デバッギングツールである。

例題演習

プログラム実行中の特定の時点で成立する変数間の関係や条件を記述した論理式を埋め込んで，そのプログラムの正当性を検証する手法はどれか。

ア　アサーションチェック　　イ　コード追跡
ウ　スナップショットダンプ　　エ　テストカバレッジ分析

解答解説

プログラムの正当性を検証する手法に関する問題である。

アのアサーションチェックはエラーのないプログラムを作るためのツールで、実行時にそれが満たされていない場合にエラーや例外を発生させたり、メッセージを表示して処理を中断したりする機能である。プログラム中のバグや不具合、論理の矛盾の発見に使われる。プログラムの正当性検証に使用する。求める答えはアとなる。

イのコード追跡は、トレーサを使用して行う。トレーサ上でプログラムを実行すると、その過程を追跡・記録し、実行された命令を順番に、レジスタや変数の値などの状態と共に表示してくれる。これを見ることで、プログラムが意図したとおりに動作しているかを調べたり、どこに誤りがあるかを発見するのが容易になる。

ウのスナップショットダンプは、プログラムにデバッグ命令を組み込んでおき、デバッグ命令を実行する都度、主記憶装置の一部やレジスタの内容を書き出す。指定した条件のときだけ主記憶やレジスタの内容などの追跡データを出力するダンプである。

エのテストカバレージ分析は、テスト対象プログラムのうち、既テスト分と未テスト分を識別するツールである。あるテストデータがプログラムのどの経路を通ったかを調べながら、プログラム全体の経路のうち約何%をカバーしたかを求めることができる。

例題演習

ホワイトボックステストにおいて，コード中のどれだけの割合の部分を実行できたかを評価するのに使うものはどれか。

ア　アサーションチェッカ　　イ　シミュレータ
ウ　静的コード解析　　エ　テストカバレージ分析

解答解説

ホワイトボックステストの支援ツールに関する問題である。

アのアサーションチェッカは、エラーのないプログラムを作るためのツールで、ある条件が成立していなければならない部分にチェック用のコードを入れ，その条件に違反している場合はエラーを出力することで，プログラムをチェックする仕組みである。

イのシミュレータは、ある計算機で実行可能な目的プログラムの命令を１つずつ解釈し、他の計算機の命令に変換するプログラムである。

ウの静的コード解析は、実行ファイルを実行することなく解析を行う。ソースコードに対して行われることが多く、オブジェクトコードに対して行う場合もある。

エのテストカバレージ分析は、ソフトウェア開発において、出来上がったプログラムのテストをする際に、どの程度をテスト対象とするかまたはテストを実行したかの割合を分析することである。求める答えはエとなる。

例題演習

結合テストの目的に関する記述として、適切なものはどれか。

ア　機能が外部設計書に記されているとおりに実現されているかどうかを検証する。
イ　処理時間や応答時間が目標を満たしているかどうかを検証する。
ウ　プログラムの部品であるモジュール間のインタフェースを検証する。
エ　目標どおりにジョブの多重度や端末の同時接続が実現できるかどうかを検証する。

解答解説

結合テストの目的に関する問題である。

結合テストは、モジュール間のインタフェースとプログラムの機能を検査する。

ア、イ、エはシステムテストである。システムテストでは、インタフェーステスト、機能テスト、性能テスト、障害テストや負荷テストが行われる。

ウは結合テストで、求める答えはウとなる。

例題演習

トップダウンテストに関する記述として，適切なものはどれか。

ア　下位のモジュールを代行するドライバの作成が必要になる。
イ　重要度の高い上位のモジュールがテストで繰り返し使用されるので，上位モジュールの信頼性が高くなる。
ウ　テストの最終段階でモジュール間のインタフェース上の問題が生じやすい。
エ　モジュール数の少ない上位部分から開発していくので，開発の初期段階からプログラミングとテストの並行作業が可能である。

解答解説

トップダウンテストに関する問題である。

トップダウンテストの最上位モジュールは最初からテストが繰り返されており、バグの発見も効率よく行え、プログラム全体の制御を司る重要な部分がきちんとテストできる。

アの下位モジュールを代行するのはドライバではなく、スタブである。

イの内容はトップダウンテストの特徴を表している。求める答えはイとなる。

ウの最終段階でモジュール間のインタフェースに問題が生じやすいのはボトムアップテストで

ある。

エの開発の初期から並行作業が可能なのはボトムアップテストである。

例題演習

ボトムアップテストの特徴として，適切なものはどれか。

ア　開発の初期の段階では，並行作業が困難である。
イ　スタブが必要である。
ウ　テスト済みの上位モジュールが必要である。
エ　ドライバが必要である。

解答解説

ボトムアップテストに関する問題である。

アの並行作業は可能である。

イはスタブではなく、ドライバが必要である。

ウはドライバがあれば、テスト済みの上位モジュールは必要ない。

エのドライバが必要は正しい。求める答はエとなる。

例題演習

テスト手法の一つであるボトムアップテストの説明として、適切なものはどれか。

ア　下位のモジュールから上位のモジュールへと順に結合しながらテストする方法であり、未完成の上位モジュールの代わりにドライバが必要である。
イ　個々のモジュールを独立にテストし、各々のテストが終了した時点ですべてを結合してテストする方法である。
ウ　上位のモジュールから下位のモジュールへと順に結合しながらテストする方法であり、未完成の下位モジュールの代わりにスタブが必要である。
エ　単体テスト、結合テスト、システムテスト、運用テストの順にテストする方法である。

解答解説

ボトムアップテストに関する問題である。

アはボトムアップテスト、イはビックバンテスト、ウはトップダウンテスト、エは通常の開発工程で順次行われる一連のテスト手順である。求める答えはアとなる。

例題演習

モジュール間やサブシステム間のインタフェースを検証するために行うテストはどれか。

ア　運用テスト　　イ　結合テスト
ウ　システムテスト　　エ　単体テスト

解答解説

結合テストに関する問題である。

アの運用テストは、ユーザ部門の運用グループが実際の運用と同一の条件を作り出し、機能面および操作面の双方のテストを行うことである。

イの結合テストは、サブシステムやモジュールを結合したテストで、サブシステムやモジュール間のインタフェースや入出力の検証テストである。求める答えはイとなる。

ウのシステムテストは、ソフトウェアやシステムが要求された仕様に合致しているかどうかを検証するために、複数の機能グループを組み合わせて行うテストである。

エの単体テストは、プログラムの最小単位であるモジュールの品質をテストすることで、モジュール内のエラーを発見することである。

例題演習

ボトムアップテストにおいて、被テストモジュールの上位のモジュール機能を代行するのはどれか。

ア　インタプリタ　　イ　コンパイラ　　ウ　スタブ　　エ　ドライバ

解答解説

ボトムアップテストにおけるテスト支援ツールであるドライバに関する問題である。

アのインタプリタは、高水準言語で書かれた実行プログラムを１行ずつ解釈しながら実行していくプログラムである。

イのコンパイラは、プログラムをコンピュータに理解できる機械語に変換するプログラムの一種である。

ウのスタブは、トップダウンテストに用いられるツールである。

エのドライバは、ボトムアップテストに用いられるツールである。求める答えはエである。

例題演習

モジュールテストで使用されるドライバ又はスタブの機能に関する記述のうち、適切なものはどれか。

ア　スタブは、テスト対象モジュールからの戻り値を表示・印刷する。
イ　スタブは、テスト対象モジュールを呼び出すモジュールである。
ウ　ドライバは、テスト対象モジュールから呼び出されるモジュールである。
エ　ドライバは、テスト対象モジュールに引数を渡して呼び出す。

解答解説

テスト支援ツールのスタブ、ドライバに関する問題である。

ドライバはテストモジュールの上位モジュールの機能をシミュレートする。テストモジュールから見て主プログラムの役割を果たす。結合テストのボトムアップテストに利用される。スタブはテストモジュールの下位モジュールの機能をシミュレートするものである。テストモジ

ュールから見て副プログラムの役割を果たす。結合テストのトップダウンテストに利用される。

ア、イはドライバの説明、ウはスタブの説明、エはドライバの説明で適切である。求める答えはエとなる。

例題演習

テスト工程におけるスタブの利用方法に関する記述として、適切なものはどれか。

ア　指定した命令が実行されるたびに、レジスタや主記憶の一部の内容を出力することによって、正しく処理が行われていることを確認する。

イ　トップダウン的にプログラムのテストを行うとき、作成したモジュールをテストするために、仮の下位モジュールを用意して動作を確認する。

ウ　プログラムの実行中、必要に応じて変数やレジスタなどの内容を検査し、必要であればその内容を修正した後、後続の処理のテストを行う。

エ　プログラムを構成するモジュールの単体テストを行うとき、そのモジュールを呼び出す仮の上位モジュールを用意して、動作を確認する。

解答解説

テスト支援ツールのスタブに関する問題である。

アはスナップショット、イはスタブの利用方法、ウはデバッガ、エはドライバである。求める答えはイとなる。

例題演習

システム開発におけるテストでは、小さな単位から大きな単位へ、テストを積み上げて行く方法がとられることが多い。このとき、テストの適切な実施順序はどれか。

ア　システムテスト→結合テスト→単体テスト

イ　システムテスト→単体テスト→結合テスト

ウ　単体テスト→結合テスト→システムテスト

エ　単体テスト→システムテスト→結合テスト

解答解説

システム開発におけるテスト順序に関する問題である。

単体テストは、最小単位であるモジュールの検証および品質をテストする。結合テストは、モジュールを結合して行うテストで、モジュール間のインタフェースの検証、プログラムの入出力を含むテストである。システムテストは、ソフトウェアやシステムが、仕様に合致しているかどうかを検証するテストである。

システム開発におけるテスト順序は、単体テスト→結合テスト→システムテストである。求める答えはウである。

例題演習

システムテスト工程で実施するテストはどれか。

ア　負荷テスト
イ　モジュール間のインタフェーステスト
ウ　モジュール仕様書に基づいた動作確認テスト
エ　レグレッションテスト

解答解説

システムテストに関する問題である。

アはシステムテストで確認する内容、イは結合テストで行う内容、ウはブラックボックステストで確認する内容、エのレグレッションテストはソフトウェアの保守時に行うテストである。求める答えはアとなる。

例題演習

運用テストの実施体制や手順に関する記述として，適切なものはどれか。

ア　運用テストは，システムテストの前工程として実施する。
イ　開発部門がテストケースを設定し，ユーザ部門がこれに従いテストをする。
ウ　開発部門の最後の責任として開発部門主導でテストをする。
エ　ユーザ部門が主体となり，実際に運用するときと同じ条件でテストをする。

解答解説

運用テストの実施体制や手順に関する問題である。

運用テストは、運用と同じ環境で実データを使って行うテストで、利用者が新システムへ業務を移行しても問題がないかどうかをテストして確認する。従って、ユーザ部門が主体となり、実際に運用するときと同じ条件でテストする。

アの運用テストの順序はシステムテストの後工程であって、前工程ではない。

イのテストケースの設定は、ユーザ部門が行うもので、開発部門が設定するものではない。

ウのテストの主導はユーザ部門が行うもので、開発部門主導のテストではない。

エのユーザ部門が主体であり、運用の条件でのテストであるという内容は運用テストの記述である。求める答えはエとなる。

例題演習

ソフトウエアのテスト方法のうち，ソフトウエア保守のために変更した箇所が他の部分に影響しないかどうかを確認する目的で行うものはどれか。

ア　運用テスト　　イ　結合テスト
ウ　システムテスト　　エ　レグレッシヨンテスト

解答解説

レグレッションテスト(退行テスト)に関する問題である。

アの運用テストは、運用と同じ環境で実データを使って行うテストで、利用者が新システムへ業務を移行しても問題がないかどうかをテストして確認する。

イの結合テストは、モジュール間のインタフェースとプログラムの機能を検査する。

ウのシステムテストは、プログラム間の結合とシステムの機能を検査する。

エのレグレッションテストは、既存のソフトウェアを修正する場合、その与える影響についてテストすることで、修正内容と関係ない部分に影響を与え、異常が発生していないかどうかを確認するテストである。求める答えはエとなる。

例題演習

システムの移行テストを実施する主要な目的はどれか。

ア　安全性・効率性の観点で，既存システムから新システムへの切替え手順や切替えに伴う問題点を確認する。

イ　既存システムのデータベースのコピーを利用して，新システムでも十分な性能が発揮できることを確認する。

ウ　既存のプログラムと新たに開発したプログラムとのインタフェースの整合性を確認する。

エ　新システムが要求されたすべての機能を満たしていることを確認する。

解答解説

移行テストに関する問題である。

システムの移行は、新システムを稼働させるために、現行システムから新システムへ、ハードウェアやソフトウェア、各種ファイルを円滑に移し変えることである。そのために、移行方法や移行手順、移行体制、移行日程計画、移行タイムチャートなどを作成する。移行作業の中で大きな比重を占めるものがデータの移行である。利用部門と協同して、現行の業務フロー、データベース仕様を元に、現行システムと新システムのデータ項目を比べ、移行対象データ項目を決める。ハードウェア、ソフトウェア、データの移行方法、移行のタイミングも重要な検討項目である。新たなシステム開発に伴って、ハードウェアの入れ替えや増強、ＯＳやミドルウェアの入れ替えや変更、通信回線の新設や増設が行われる。

アの安全性・効率性の観点で、既存システムから新システムへの切り替え手順、切り替えに伴う問題点を確認するのは移行テストの内容に含まれる。求める答はアとなる。

イのデータベースの確認は運用システムの運用効率・業務効率の測定で行う作業である。

ウのテストは結合テスト、システムテストで実施する。

エの機能確認テストはシステムテストで実施する。

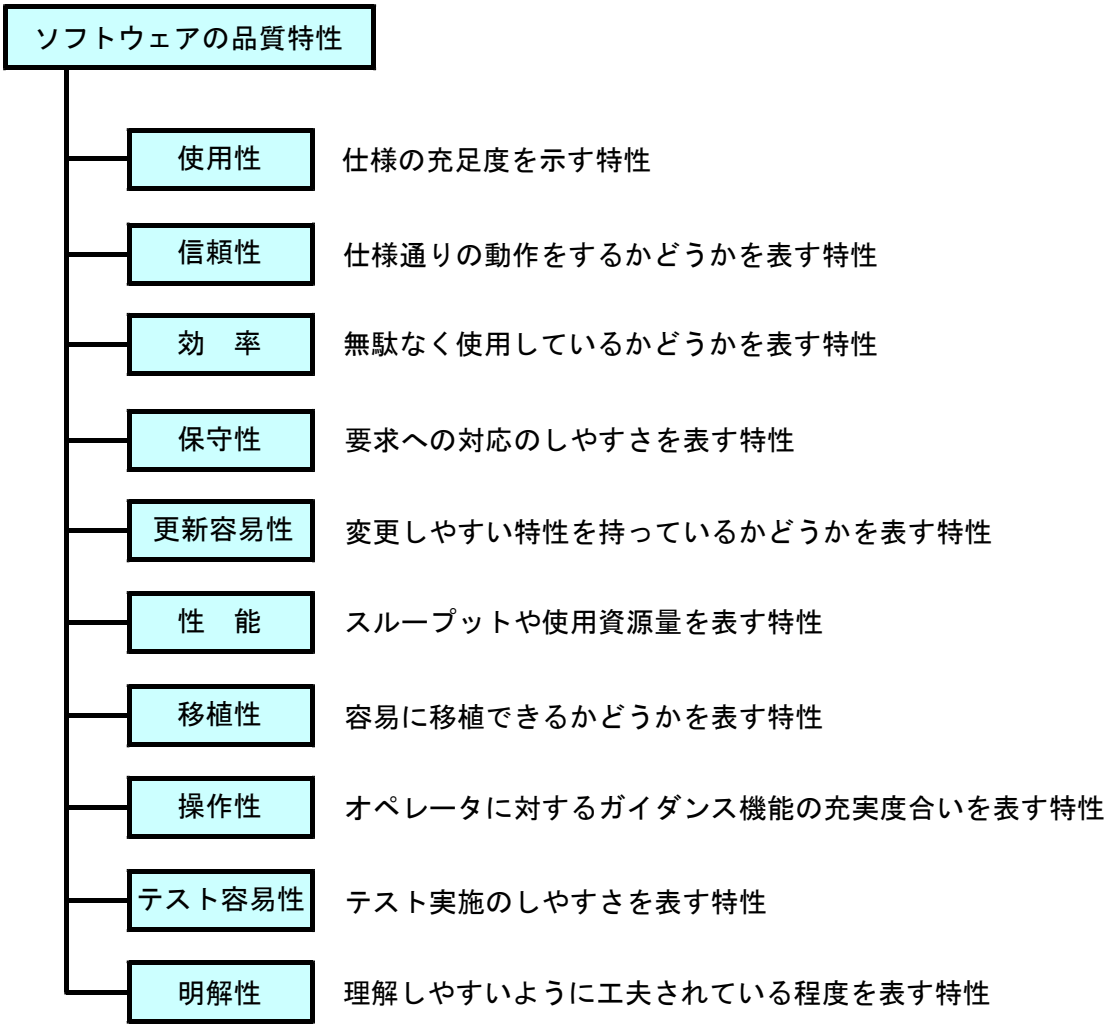
txt04041　開発・運用管理

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

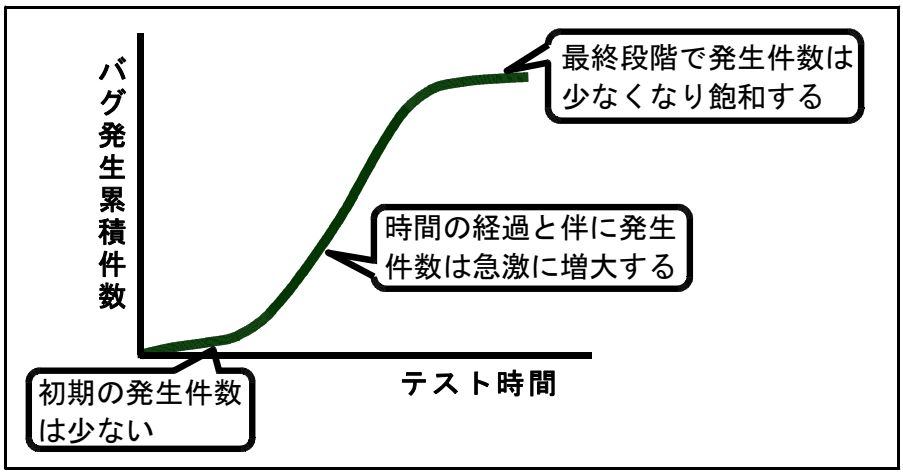
㋒　意図的なエラーＳ個全部が検出されるまでテストを継続する。

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いう判断し、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。

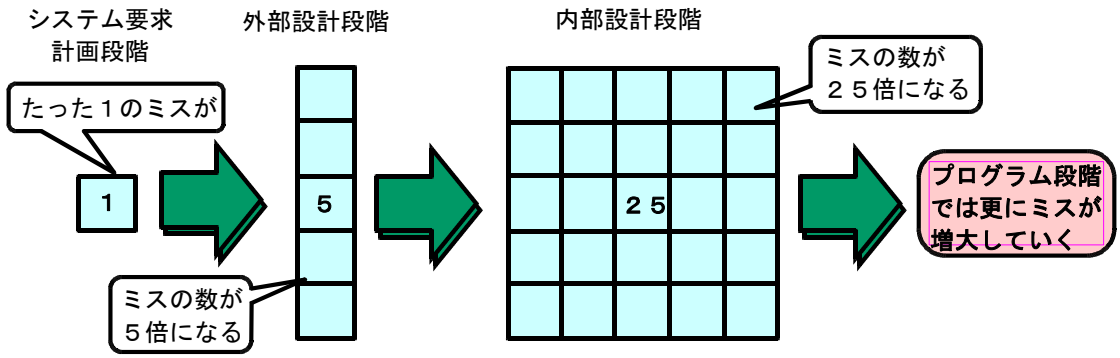
ⓒ　デザインレビューの作業内容

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。

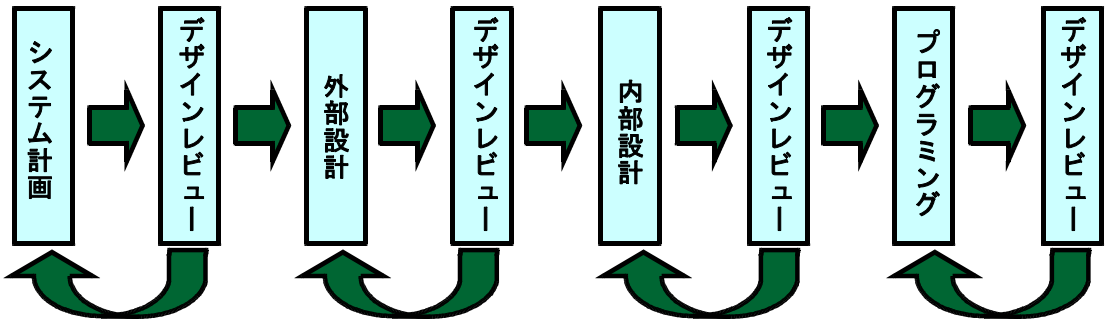
㊓　品質状況の評価結果を分析し、今後の改善施策へフィードバックする。

ⓓ　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

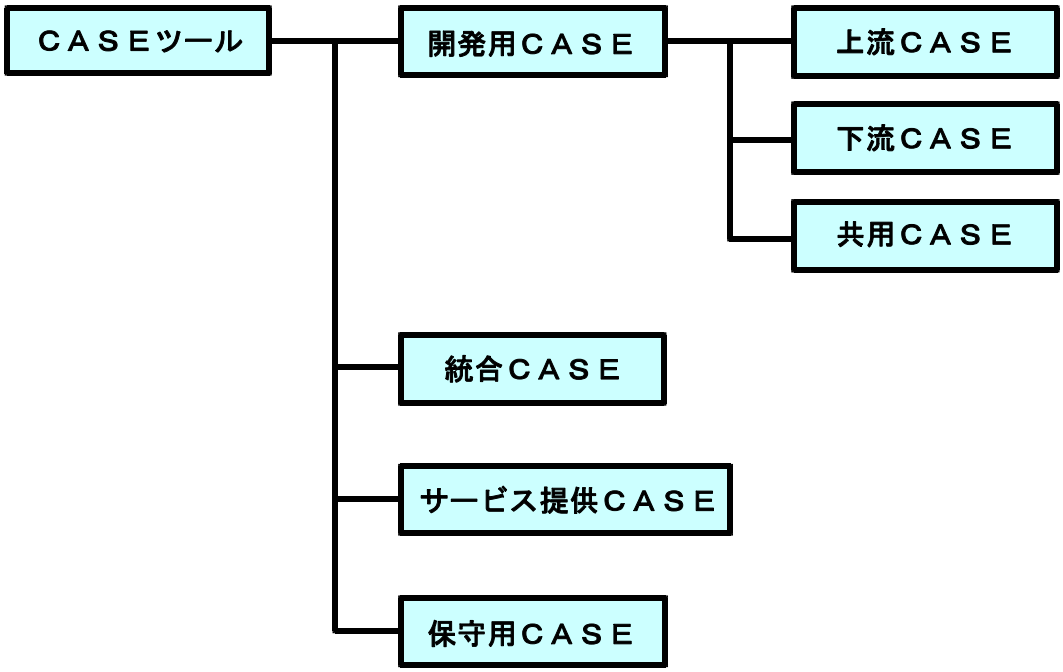
ⓐ　ＣＡＳＥ

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

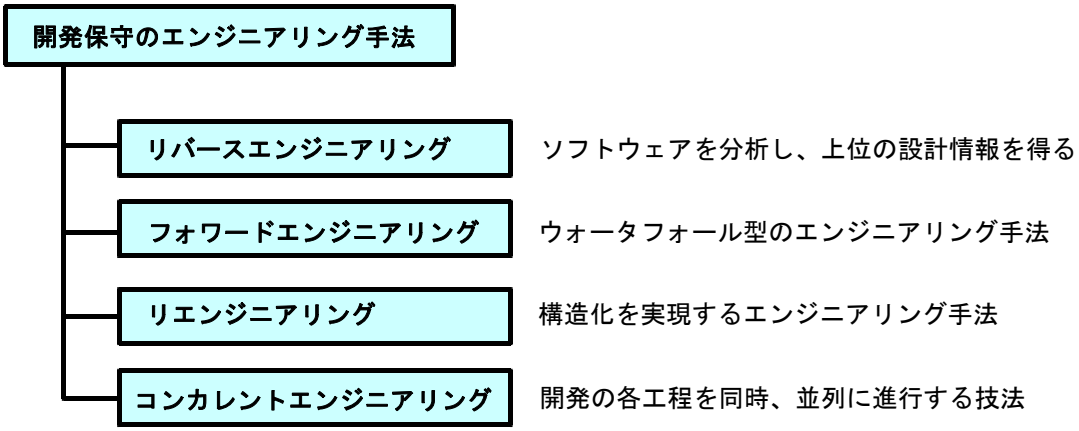
㋐　原始プログラムの解析、修正および再構造化機能

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。

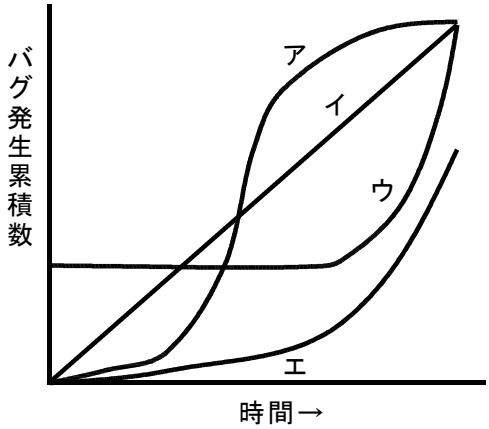
イの使用性は、仕様の充足度を示す特性で、ソフトウェアの目的とした機能が、仕様通りに表現されているかどうかを表す特性要因である。

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

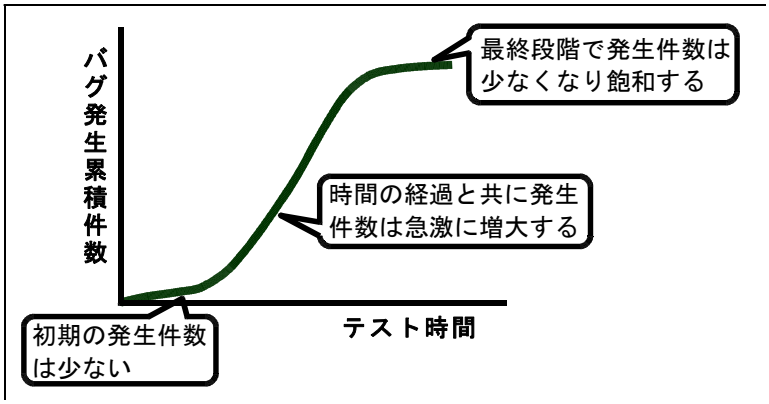
ソフトウェアの開発過程における、テストの進捗とバグ発生累積数の関係を示すグラフとして、最も一般的なものはどれか。ただし、横軸はテストの時間の経過、縦軸はバグ発生累積数である。テスト項目は適切かつ十分に設計されているものとする。



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。

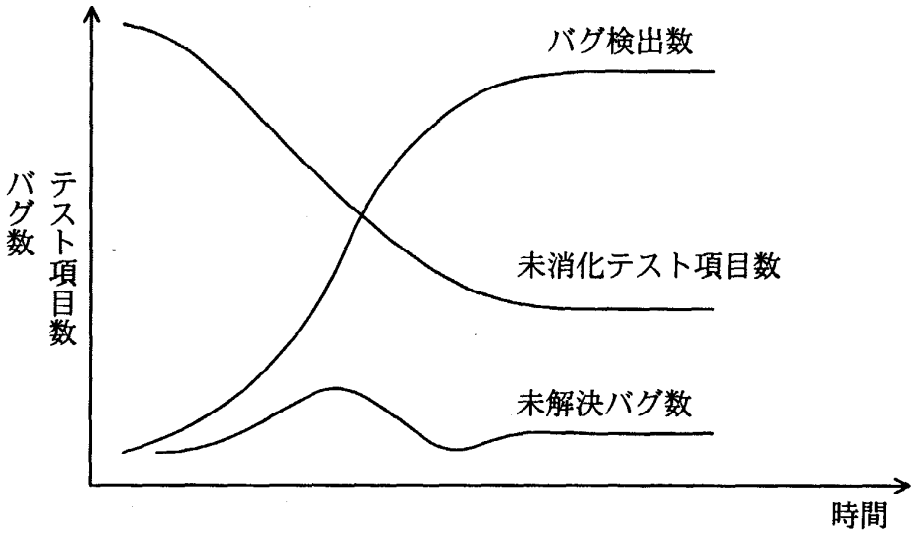
バグの総数は２２×（４８÷１８）＝６６

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。

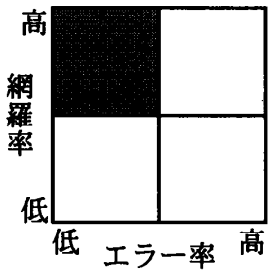
イのテスト項目の消化実績が上がっているは、②の現象が一致しない。

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。

txt040412 開発組織と開発管理

① 開発組織と体制

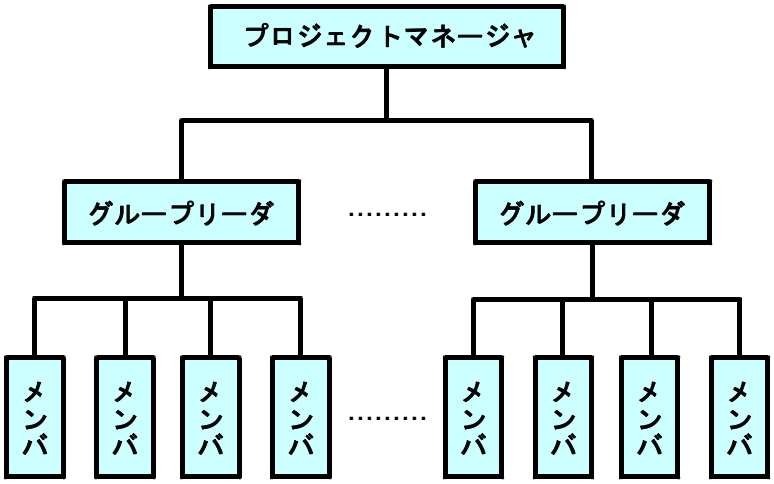
ⓐ プロジェクト組織

プロジェクト組織は、プロジェクトの遂行を目的とする一時的な組織である。システム開発では、システム化計画が承認されるとプロジェクトチームが編成され、開発組織体制ができる。

プロジェクトの目的、規模、期間、活動内容や展開要領に対応して組織体制が異なってくる。システム開発における組織体制の代表的なものに、階層型チームとチーフプログラムチームがある。

ⓑ 階層型チーム

プロジェクトマネージャ１名と数人のプロジェクトリーダを置き、メンバで構成されるチームである。



ⓒ 階層型の特徴

㋐ システム開発での一般的なチーム編成である。
㋑ 比較的大規模なシステムに採用される。
㋒ コミュニケーションの良否が比較的問題になる。

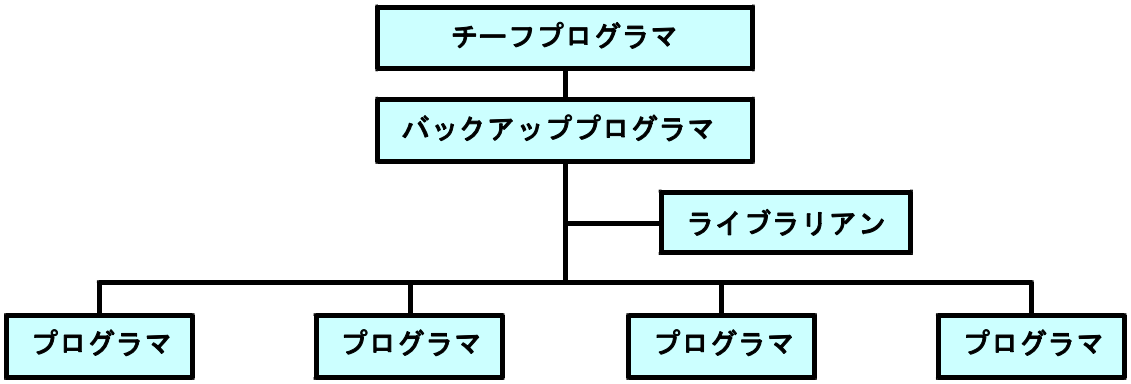
ⓓ 階層型組織の役割

㋐ マネージャは、プロジェクトチームの最高責任者で、グループ間の調整や設計・プログラミングなどの方針決定、チーム管理などのプロジェクト管理を担当する。
㋑ グループリーダは、マネージャの補佐とメンバのバックアップを行い、担当グループの責任者として作業をとりまとめる。

㋒　メンバは、外部設計、内部設計、プログラム設計、プログラミング、テスト、文書作成などの実務作業を行う。

ⓔ　チーフプログラマチーム

比較的少人数(10名程度まで)で構成されるプロジェクトチームで、チーフプログラマの全責任のもとに作業を明確に分担して、チーム内のコミュニケーションを円滑にし、プログラムの生産性や品質の向上を図る方式である。



ⓕ　チーフプログラマチームの特徴

㋐　比較的小規模のプロジェクトで採用される。
㋑　バックアッププログラマとライブラリアンが存在する。
㋒　リーダの負担は大きいが、リーダの育成に適している。
㋓　プログラマのモラルの低下を招きやすい。

ⓖ　チーフプログラマチームの組織の役割

㋐　チーフプログラマ

チーフプログラマは、プログラム作成の全責任をもち、技術や管理の責任者である。チームの統率、システムの最重要モジュールの設計と開発を担当する。

㋑　バックアッププログラマ

バックアッププログラマは、チーフプログラマの支援を行う。チーフプログラマと同程度の能力の持ち主で、チーフプログラマの代行も行う。

㋒　ライブラリアン

ライブラリアンは、プログラミング以外の事務作業、文書やプログラムの保守管理を行う。

㋓　プログラマ

プログラマは、チーフプログラマやバックアッププログラマによって設計された部分のプログラミングを行う。

② ソフトウェアライフサイクルプロセス（ＳＬＣＰ）

ⓐ ライフサイクルコスト

システムの開発、運用、保守で、積み重なっていくコストであり、新しいシステム開発を計画する場合、開発コストと共にライフサイクルコストを評価する必要がある。

ⓑ ライフサイクルモデル（ＳＤＬＣ）

ソフトウェアの誕生から消滅に至るＳＤＬＣ(Software Development LifeCycle)として採用したプロセスモデルである。システム開発の準備段階で開発のプロセスモデルを選択し、ＳＤＬＣモデルを決定する。

ⓒ ＳＬＣＰ－ＪＣＦ

ＳＬＣＰ－ＪＣＦは、コンピュータ・システムの開発において、システム発注側(ユーザー)と受注側(ベンダ)の間で相互の役割や業務の範囲・内容、契約上の責任などに対する誤解がないように、双方に共通して利用できるように用語や作業内容を標準化するために作られたガイドラインであり、ソフトウェア開発と取引のための共通の枠組みについて規定している日本国内規格である。

システム構築・運用の受発注において、契約上のトラブル防止、作業内容の確認、役割分担の明確化、社内作業標準の策定や人員計画、見積もり精度の向上、品質確保などに利用される。

作業内容をプロセス、アクティビティ、タスクに階層化して定義し、プロセスは取得・供給・契約変更管理・企画・要件定義・開発・運用・保守という基本業務の８プロセスと支援系(文書化、品質管理など)の９プロセス、組織、システム監査と修正のプロセスを加えた構成になっている。

③ 能力成熟度モデル統合

ⓐ 能力成熟度モデル統合（ＣＭＭＩ）

ＣＭＭＩは、システム開発を行う組織がプロセス改善を行うためのガイドラインである。

成熟モデルはソフトウェア開発の現状を評価するための手段として、製品、プロセス、資源などについて評価する。組織が出荷しサポートしている製品、製品を開発したプロセスを評価するために、製品開発プロセス、分析設計手法、テストと文章化技法、変更管理方針などを調査する。組織が所有しているソフトウェア開発環境、チーム、コンピュータハードウェア、ソフトウェアツールからなる資源が評価対象になる。

ⓑ 成熟モデルの評価点

㋐ 見積や期日の正確さ

㋑　**下請企業の管理能力**

㋒　**不具合回避の実績**

㋓　**企業の開発能力**

ⓒ　成熟モデルの評価レベル

組織に対する評価レベルをレベル１～レベル５に設定して、企業や対象の組織能力を評価する。上位の成熟度レベルにあると評価された組織は、先進的なソフトウェア開発の能力を持っていることになる。

㋐　レベル１

プロジェクトやソフトウェアの管理が場当たり的で、開発の成否を個人の能力や努力に依存しており、開発の正式な管理がされていない。

㋑　レベル２

コスト、日程などのプロジェクト目標や管理手順が明確で、過去の成功事例を繰り返せことができ、品質保証のためのメカニズムが確立している。

㋒　レベル３

開発目標や管理手順を組織で共有し、常に改善を目指すプログラムを確立している。

㋓　レベル４

開発プロセスや商品の品質、目標を定量的に分析し、必要に応じた変更を即座に行える。

㋔　レベル５

新しいアイデアや技術を駆使した実験データを常に定量的に分析し、危機を予測した組織的な改善が自発的、継続的に行われている。

④　プロジェクト管理

ⓐ　開発プロジェクト

プロジェクトは、特定の目的や対象・目標をもって実行される定常業務活動でない有期性の活動である。プロジェクトには、明確な開始時期と終了時期があり、期限がある活動である。プロジェクトの遂行やその成果物など基本的な部分で一度限りの活動となる。システム開発はプロジェクトとして行われ、限定された期間だけの活動組織で、通常、１～５年ぐらいの期間の場合が多い。

ⓑ　プロジェクトの発生理由

㋐　市場の要求

ある市場が成果物を要求し、それを満たそうとする時

㋑　ビジネス・ニーズ

組織の中で成果物を要求し、それを満たそうとする時

㋒　顧客要求

取引先の外部から成果物の要求があり、それを満たそうとする時

㋓　技術の進歩

技術の進歩に伴い、新しい技術を利用した成果物を開発しようとする時

㋔　法的要求

法律などの制定に伴い、それに対応しようとする時

㋕　社会的ニーズ

自治体や町内など、社会生活単位での要求があり、それを満たそうとする時

ⓒ　プロジェクト管理とは

プロジェクトの目的を完遂またはそれ以上の成果を上げるためには、最適な知識やツール、技法を駆使してプロジェクトを遂行することである。ソフトウェア開発では、開発の背景、目的、範囲を明確にした上で、開発期間、開発費用、品質管理、開発組織体制、コミュニケーション計画、リスク管理、調達管理を構成要素として管理体系を決定する。プロジェクトのフェーズごとに作業内容、要員、工数などの資源割り当てを定め、品質などを把握しながらコントロールしていく。

ⓓ　プロジェクトマネージャ

プロジェクト管理を行う責任者であり、プロジェクト計画の立案、ＰＤＣＡサイクルによるプロジェクトの管理・遂行、プロジェクトの利害関係者との調整、クライアントやプロジェクトメンバおよび関連部門とのコミュニケーション、メンバのモチベーションの維持・向上などがある。

⑤　プロジェクト・スコープ管理

ⓐ　プロジェクトの成功とは

プロジェクトが成功したということは、プロジェクトに課せられた制約事項を満たし、プロジェクトの目標が達成されたということである。プロジェクトの主要制約事項に、スコープ、タイム、コストの３つがある。

ⓑ　プロジェクトの制約事項

㋐　スコープ

スコープはプロジェクトの作業範囲のことである。プロジェクトにおけるスコープには、プロジェクトの成果物に求められる成果物スコープ、成果物を作成するために必要な作業を示すプロジェクトスコープがある。プロジェクトスコープマネジメントでは後者のプロジェクトスコープが対象となる。

㋑　タイム

タイムはプロジェクトの成果物を完成させる時期のことである。プロジェクトタイムコントロールでは、プロジェクトの所要期間を見積、スケジュールを作成し、スケジュールコントロールを行います。

㋒　コスト

コストはプロジェクトの目標を達成するのに費やす費用のことである。プロジェクトコストマネージメントでは、コストを見積、予算化し、コストコントロールを行う。

ⓒ　プロジェクトの範囲

プロジェクトの範囲はプロジェクト実施対象と役務範囲のことであり、プロジェクト管理では最初に決定しておく事項である。プロジェクトをその目的に合わせて遂行するためには、その活動の対象を明確にしておく必要がある。

ソフトウェア開発では、開発対象業務、開発対象部署、プロジェクト成果物、提供するサービス、利用するツールなどを規定し、プロジェクトの範囲を逸脱しないように管理する。

⑥　ステークホルダー

ⓐ　ステークホルダーとは

ステークホルダーとは、プロジェクトの実行や完成によって自らの利益に影響が出る人や組織のことである。

ソフトウェア開発プロセスに関係してくるステークホルダーには次のものがある。

㋐　上級管理者
㋑　プロジェクトマネジャー
㋒　開発者
㋓　顧客
㋔　ユーザ

ⓑ　ステークホルダーの役割

プロジェクトには複数の人が関与し、立場によって、プロジェクトに関係する役割が異なる。主要な役割として次の事項がある。

㋐　プロジェクトの目標に向けて作業を進める。
㋑　プロジェクトに必要な資金を提供する。
㋒　プロジェクトの成果物を利用する。
㋓　作業実施以外の面で作業を支援する。
㋔　プロジェクトに関係する様々な要望を提出する。

⑦　プロジェクトコスト管理

ⓐ　コスト管理のプロセス

プロジェクトのコスト管理には、コスト見積、コストの予算化、コストコントロールの３つのプロセスがある。

㋐　コスト見積

プロジェクトが完了するまでに必要な資源コストを算出する。プロジェクトの開発環境や過去のプロジェクトの経験などを参考にして、各活動のコストを見積もる。

㋑　コストの予算化

コストの予算化はプロジェクトの個々の作業の成果を測定するコストベースラインを設定する目的で行われる。活動毎のコスト見積、スケジュールなどをもとに、成果物毎、期間ごとに発生するコストを集約する。コストベースラインは必要となるコストを時系列的に展開したもので、進捗管理を行う上での重要な基準になる。

㋒　コストコントロール

コストベースラインと実コストのずれを把握し、コストが超過しないように処置をとる。

ⓑ　コストの見積法

㋐　類推見積

過去の類似プロジェクトの実コストを参考にコストを見積もる方法である。

㋑　ボトムアップ見積

最も詳細レベルの活動のコストを見積、それを集計し、総額を算出する。

㋒　係数見積

過去のデータをもとに、実コストとそれに影響を及ぼす変数との関係を統計処理し、コスト見積の算出式を求める。この算出式をもとにコストを見積もる。

⑧　ソフトウェアコストの見積

ⓐ　ソフトウェアコスト

ソフトウェアコストはソフトウェア開発のライフサイクルの各フェーズで必要とされる費用である。システムを開発する場合、システムのソフトウェア開発に対するコストを見積もる必要がある。見積を行うためには、全フェーズに対して見積工数を算出しなければならない。

ⓑ　コストの見積の手順

㋐　開発プロセスモデルの選択
㋑　開発方法の選択
㋒　開発環境の設定
㋓　開発上の制約の明確化
㋔　見積対象フェーズの明確化
㋕　見積手法の選定
㋖　見積の実施

ⓒ　類似プロジェクトに基づく見積法

過去の類似システムの実績値をもとにシステムの開発工数を見積もる手法である。過去に開発したシステムの内、特性や処理内容、規模などが類似したものの実績データを使用する。

類似システムが存在するなど、条件が一致すれば高い見積もり精度が得られる。しかし、開発条件が不明確であったり、過去のプロジェクトの条件と一致しない場合は適用できない。開発対象システムが類似していても、開発環境や状況が異なると適用が難しい。

ⓓ　類似プロジェクトに基づく見積法の手順

㋐　データの収集

開発が完了したプロジェクトのデータを収集し、プロジェクトの特性や規模、開発のフェーズ、所要工数、開発期間、開発要員、生産性などを整理する。

㋑　見積対象プロジェクトの分析

業種や業務処理分野、企業規模、開発に使用するプログラム言語、見積のフェーズ、プロジェクトの規模などのプロジェクトの特性を分析する。

㋒　見積の実施

分析したプロジェクトの特性に最も近いプロジェクトを、収集したプロジェクトの中から探し、類似したプロジェクトのデータを参考にして、対象プロジェクトの見積を行う。工数には人月を利用する。工数の見積もり時には、開発規模、開発の生産性が重要な情報になる。

ⓔ　経験的見積もり法

開発経験者が過去のプロジェクト経験からシステムの開発規模を見積もる手法である。開発経験者が見積もれば高い精度が得られるが、属人性が強い見積もりになったり、条件が異なると適用できない難点がある。

⑨　開発コストモデル

ⓐ　ファンクションポイントモデル(ＦＰ法)

ＦＰ法は1983年にAlbrechtによって提案されたコストモデルで、ソフトウェアに含まれる機能単位に見積もる方法である。評価対象の領域を明確にし、システムの提供する５つの機能(外部入力・外部出力・外部照会・内部論理ファイル・外部インタフェース)を抽出し、それぞれのデータタイプに対して、単純レベル、多少複雑レベル、複雑レベルの重み付けを加味して点数をつけ計量化する。

ファンクションポイントＦＰは次の式で計算する。

$$ＦＰ＝Ｃ×(Ｐ_1＋Ｐ_2＋Ｐ_3＋Ｐ_4＋Ｐ_5)$$

画面や帳票等の入出力インタフェースの多い場合に適している。外部入出力や内部論理ファイル、照会、インタフェースなどの個数や特性などから開発規模を見積もる。

ⓑ　ＣＯＣＯＭＯ

1981年にBoehmにより提案されたソフトウェアのコストモデルで、システムの特性や開発環境を考慮し、プログラマの作業量を統計的なモデルによって計算式で算定する方法で、中、大規模のシステム開発に適用できるモデルある。ＣＯＣＯＭＯはウォータフォールモデルを前提にし、ＣＯＣＯＭＯⅡはスパイラルモデルに対応している。

ソフトウェアプロジェクトを３種類に分類して名目工数を見積もり、これに、製品、計算機、人的、プロジェクト的属性による乗数を掛けて最終値を得る。プログラムの作業工数Ｐは次の式で計算する。

$$Ｐ＝ｘ×(ＫＤＳＩ)^{ｙ}×Ｚ$$

ＫＤＳＩ：プログラムの大きさを示す。注釈行を除き、ＪＣＬを加えた総数である。

ｘ、ｙ、Ｚ：定数でいくつかの尺度基準で決まる。尺度基準には、見積対象レベル(概要、通常、詳細)、作業要員の構成を考慮する。開発環境、開発要員の能力に依存する。

開発規模の調整要素として再利用に関する項目や生産性に与える要因があるのが特徴である。

開発期間　$Ｔ＝2.5×Ｐ^{ｙ}$ で求められる。

ⓒ　ＬＯＣ法

ＬＯＣはプログラムのソースコード行数のことであり、システムに実装する処理内容などをプログラム言語でコード化したものの行数を表す。システム規模の指標として利用する。ＬＯＣ法は、ソースコードを基準として開発工数を見積もる方法である。開発環境のプラットフォームや使用言語が固定という前提で効果的な手法である。クライアントサーバシステムやオブジェクト指向技術などの複合的な開発環境では使用に限界がある。

⑩　開発工程の管理

ⓐ　工程管理

システム開発プロジェクトの目標を達成するためには、適切な計画・スケジュールを工程表で作成し、それに基づいた進捗管理を行うことが重要である。工程表の表現法にはガントチャートとＰＥＲＴの方法がある。

進捗管理の目的はプロジェクトが計画通りに進捗しているかどうかを把握し、問題を早期に見つけ、適切な対応策を立てて実施し、プロジェクトの目標を達成することである。進捗管理は適切な開発計画を前提にした、実現可能性の高い開発計画の作成が必要である。

ⓑ　日程計画

日程計画は、プロジェクトを進める上で、タスクの所要日数や時間などを見積り、その進捗を管理する表を作成することである。日程計画表の種類には大日程計画表、中日程計画表、小日程計画表などがある。

マスタスケジュールである大日程計画表は全期間にわたる実施作業項目・工程を管理するための表である。中日程計画表は大日程計画表を工程またはサブシステムごとに分割した表である。小日程計画表は中日程計画表にある工程を更に細分化した単位で、１ヶ月や１～２週間の期間、担当者別作業項目別に週毎の進捗管理を行う。個人別の管理をすることで、作業の遅れやその原因を特定することも可能である。

ⓒ　所要期間の見積法

㋐　類推見積

過去の類似プロジェクトの実所要期間を参考に所要期間を見積る方法である。類似性が高い場合は信頼性が高い見積となる。

㋑　係数見積

作業量に生産性を掛け、基準値を算出する。総資源量に単位期間当たりの作業時間を掛け、単位期間当たりの投入資源量を算出する。基準値を単位期間当たりの投入資源量で割ることにより、所要期間を算出する。

㋒　三点見積

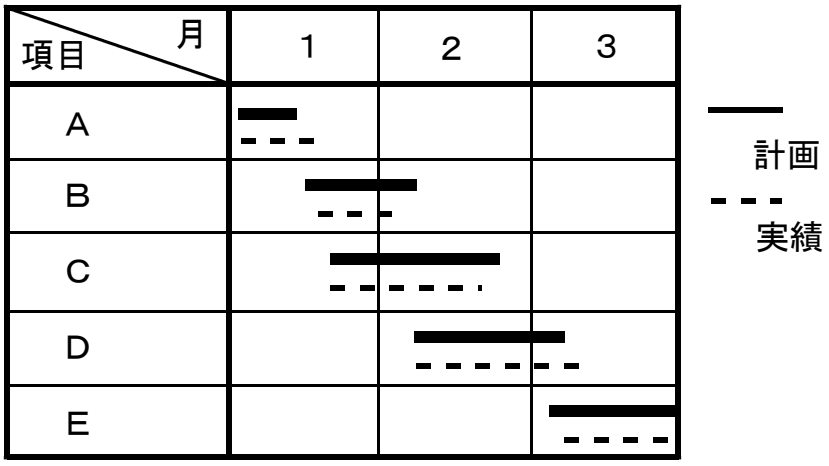
見積にリスクを加味することで、より正確な所要期間を算出する方法である。所要期間を通常に見積もりした値（最頻値）と最良のシナリオで作業が進むことを想定した見積値（楽観値）、最悪のシナリオで作業が進むことを想定した見積値（悲観値）の３種類の見積値に、重み付けを行い平均し算出する。

ⓓ　工程表の作成と活用要領

㋐　システムの規模や複雑度を見積り、開発に必要な資源を見積る必要がある。
㋑　適切な見積りに基づいて、開発環境や条件、作業の前後関係や依存関係などを考慮してスケジュールを作成する。
㋒　適切なスケジュールに基づいて、作業を進め、計画と実績を対比し、評価する。
㋓　差異があれば原因を究明して、対応策を検討し適切な措置を施す。

ⓔ　ガントチャート

ガントチャート



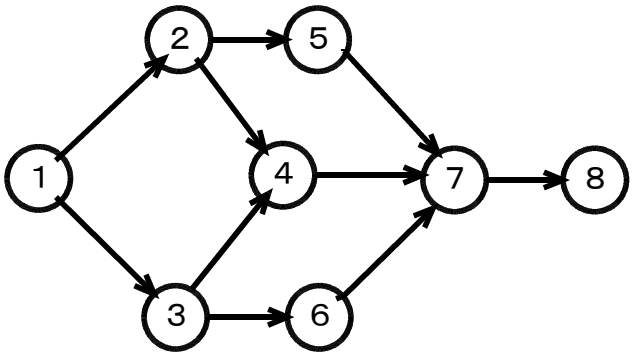
ガントチャートはプロジェクトの進捗管理のために、時間軸に対して作業項目ごとに開始から終了までを線又は帯で表した図解技法である。

ⓕ　ガントチャートの特徴

㋐　作業の開始・終了時点、現在の作業状況は明確である。
㋑　計画と実績の対比が明確である。
㋒　作業の順序関係は不明である。
㋓　作業遅れなどによる他の作業への影響度は不明である。

ⓖ　ＰＥＲＴ図

ＰＥＲＴ図



ＰＥＲＴ図はプロジェクト管理技法の一つで、時間とコストを考慮しながらプロジェクトの日程を計画し、管理する技法である。矢印線で活動、活動の結合点を丸印で表すアローダイヤグラムを用いて表示する。工程総時間の管理、クリティカルパスの管理、重複作業の発見、資源の分担計画などを行う。

ＰＥＲＴ図を作成する場合には、事前に、工数の見積り、製品機能の展開、適切なプロセスモデルの選択、プロジェクトの種類およびタスクセットの選択を行い、それらの情報を用いて進める。ＷＢＳというタスクが製品の全体または個々の機能について定義する。

ⓗ　ＰＥＲＴ図の特徴

㋐　大規模かつ複雑なシステム開発プロジェクトに対応できる。
㋑　プロジェクトのクリティカルパスや総所要日数は計算できる。
㋒　作業順序は明確である。
㋓　作業の余裕日数は計算できる。
㋔　開発コストの削減や作業日数の短縮計算にも利用できる。
㋕　各作業の所要日数の図表的表現が難しい。
㋖　計画と実績の対比が困難である。

⑪　進捗管理

ⓐ　進捗管理とは

進捗管理は、システム開発で作業工程が計画通りに進行しているかを確認することである。ガントチャートやＰＥＲＴ図を使用して行う。あらかじめ設定した管理項目に対して終了した項目の割合で進捗を計る消化率管理を用いる。進捗管理では、正確な進捗状況を把握することが重要であり、そのためには定期的にかつ正確な進捗状況を収集することが求められる。

ⓑ　進捗管理の要領

㋐　進捗度を客観的に把握する管理項目と測定尺度を決める。

㋑　進捗状況の把握の方法、進捗把握の周期、定期的なデータの収集、報告書の作成、問題の把握などについて検討し、実行する。

㋒　収集した進捗データを集計して、計画値と対比して分析する。

㋓　表面上の問題と真の問題を区分し、真の問題を探り出し、真の問題の真の原因を取り除くための対応策を検討する。

㋔　仕様変更や機能追加に関する問題、開発環境不備に関する問題、開発要員の技能不足や要員不足に関する問題、コミュニケーション不足に関する問題などが真の問題になりやすい。上司への報告と協議が必要になる。

ⓒ　工数

工数はある課題を達成するのに必要とする作業量のことである。ＩＴ業界では、作業に従事する人間の数に、必要とする時間を乗じたものを用いる。「人月」(1人が1ヶ月従事する作業が1人月)や「人日」などの単位で表す。

工数を作業者数で割ると作業時間となり、逆に、工数を時間で割ると、必要な作業者数になる。24人月の仕事を４か月で行うには、作業者は６人必要であり、逆に、８人で行うと、３か月で完成する。

工数計算は、標準的スキルを持つエンジニアが単位時間(1カ月)に行う仕事量を基準に計算するが、ソフトウェア開発においてはそれを定義する一般的な基準が存在しないため、同じ１人月でも開発ベンダによって実際に可能な仕事量は異なる。

工数は、プロジェクトの大きさを表現する場合や1人当たりの費用月額(人月単価)を掛け合わせてシステム開発費用を見積もる場合、プロジェクトの進捗管理、企業間の取引、製造部門の原価管理などに使用される。工数と実際の作業時間の差は製造部門の赤字・黒字、さらには企業の収益となって現れる。工数と実際の作業時間の比率である消化率を使用して、製造現場の管理者は管理を行う。

ⓓ　ＷＢＳ

ＷＢＳは、プロジェクトマネジメントで計画を立てる際に用いられる手法の一つで、プロジェクトをできるだけ細かい単位に分解し、管理する手法である。分割する場合、全体を大きな単位に分割してから、それぞれの部分についてより細かい単位に分割していき、階層的に構造化していくトップダウンアプローチで進める。細分化が終わったら、それぞれの部分を構成するのに必要な作業を考え、最下層に配置していく。個々の部分を構成する一連の作業のかたまりのことを「ワークパッケージ」と呼ぶ。それぞれのワークパッケージに担当する人員を配置していけば、プロジェクトを遂行する組織図ができる。ＷＢＳを用いると、プロジェクトの全体と細部が明確になり、作業管理が容易になる特徴がある。

⑫　調達マネジメント

ⓐ　調達マネージメントとは

プロジェクトでは、要素成果物を作成するのに外部からプロダクト、サービスを購入する。

調達マネージメントでは、要素成果物を外部に作成依頼するか、プロジェクト内部で作成するかを検討し、外部から購入する場合、次の６プロセスを定義し、実行する。

㋐　購入取得計画
㋑　契約計画
㋒　納入者回答依頼
㋓　納入者選定
㋔　契約管理
㋕　契約終結

ⓑ　調達プロセス

調達プロセスは次の手順で実施する。

㋐　提案評価方法の決定
㋑　提案依頼書の発行
㋒　提案評価
㋓　調達先の選定
㋔　調達の実施

ⓒ　ＲＦＩとＲＦＰ

㋐　ＲＦＩ（情報提供依頼書）

ＲＦＩは企業が調達や業務委託を行う際、自社の要求を取りまとめるための基礎資料として、外部業者に情報の提供を要請することである。あるいはその要請をまとめた文書をいう。

㋑　ＲＦＰ（提案依頼書）

ＲＦＰは情報システムの導入や業務委託を行うにあたり、発注先候補の業者に具体的な提案を依頼する文書のことである。必要なシステムの概要や構成要件、調達条件が記述されている。

ⓓ　ＲＦＩの役割

ＲＦＩの目的は、知りたい情報を文書として明確化し、外部の関係者から明快な回答を確実に得ることである。複数業者の回答を比較する場合も同一の質問に対する答えであれば、検討しやすい。ＲＦＩの発行は、提案や見積もりといった交渉の前段階で行われるもので、数多く

の外部事業者から、自社が求める能力を持ち、交渉に足る相手を絞り込むといったことも狙いとなる。ハイテクを対象としたＲＦＩでは、自社が知らない新製品・新技術に関する知見を得ることも目的となる。ＩＴ調達におけるＲＦＩはＩＴベンダの技術要件を確認するもので、どのような技術を保有しているか、どのような経験があるかなどを確認するものとなる。

ⓔ　ＲＦＰの役割

ＲＦＰは、必要とするハードウェアやソフトウェア、サービスなどのシステムの概要や、依頼事項、保証用件、契約事項などが記述されており、業者はこれをもとに提案書を作成する。発注元は業者の提案書を評価し、契約する業者を選定、ハードウェアやソフトウェア、サービスなどを調達する。

ⓕ　契約タイプ

㋐　定額契約または一括請負契約

要素成果物の取得に対して請負金額を決め、要素成果物が取得できた時に対価として請負金額を払う契約である。購入者は費用の変動リスクを納入者に転嫁できるが、その分請負金額が上がる。

㋑　実費償還契約

成果物の取得にかかる費用に、納入者の利益相当分を加えた金額を払う契約である。購入者は契約時に総額が不明確であるというリスクを負う。

㋒　タイム・アンド・マテリアル契約

購入者が支払う金額は、単位時間当たりのレートに作業に費やした時間を乗じて求めた金額とする契約である。定額契約と実費償還契約の中間に位置づけられる。

例題演習

プロジェクトの特性はどれか。

ア　独自性はあるが，有期性がない。　　イ　独自性はないが，有期性がある。
ウ　独自性も有期性もある。　　エ　独自性も有期性もない。

解答解説

プロジェクトに関する問題である。

ビジネスの業務は、大別して定常業務とプロジェクトがある。プロジェクトの特徴は有期性と独自性にある。有期性は、明確な開始と終了があることで、プロジェクトが終了するのは、目的やコールが達成された場合、何らかの理由で中止された場合のいずれかである。プロジェクトの期間は数十年にわたるもの、数週間で終わるもの様々である。独自性は、今まで存在し

なかった製品やサービスを創造することである。世界にない高速のジェット機を設計、開発することや、特定の顧客のニーズに合ったコンピュータシステムの構築などがある。

プロジェクトの特性は、独自性や有期性があることである。求める答えはウとなる。

例題演習

W．ハンフリーによって提唱されたプロセス成熟度モデル（CMM）の説明として、適切なものはどれか。

ア　ソフトウェア開発プロジェクトのプロセスを評価するためのモデルである。
イ　ソフトウェア開発プロセスモデルの一種である。
ウ　ソフトウェアプロセスの枠組みを標準化したシステム開発取引の共通フレームである。
エ　プロジェクトの成熟度に応じてソフトウェア開発の手順を定義したモデルである。

解答解説

プロセス成熟度モデルに関する問題である。

プロセス成熟度モデルは、ソフトウェア開発の現状を評価するための手段で、製品、プロセス、資源について評価する。組織が出荷しサポートしている製品、製品を開発したプロセスを評価する。評価するために、製品開発プロセス、分析設計手法、テストと文章化技法、変更管理方針などを調査する。各組織は、ソフトウェア開発環境、チーム、コンピュータハードウェア、ソフトウェアツールからなる資源を有している。これらの資源も評価対象になる。ソフトウェア評価では組織プロセスがレベル１からレベル５のどの成熟度レベルにあるかが評価される。上位の成熟度レベルにあると評価された組織は、先進的なソフトウェア開発の能力を持っていることになる。

アのソフトウェア開発プロジェクトのプロセスを評価するためのモデルが適切な記述である。求める答えはアとなる。

イは開発プロセスモデルではなく、開発プロセスの評価モデルである。

ウのシステム開発取引の共通フレームは誤りである。

エのソフトウェア開発の手順を定義したモデルは誤りである。

例題演習

CMMIを説明したものはどれか。

ア　ソフトウェア開発組織及びプロジェクトのプロセスの成熟度を評価するためのモデルである。
イ　ソフトウェア開発のプロセスモデルの一種である。
ウ　ソフトウェアを中心としたシステム開発及び取引のための共通フレームのことである。
エ　プロジェクトの成熟度に応じてソフトウェア開発の手順を定義したモデルである。

解答解説

CMMIに関する問題である。

プロセス成熟モデル(ＣＭＭ)は、ソフトウェア開発の現状を評価するための手段で、製品、プロセス、資源などについて評価する。組織が出荷しサポートしている製品、製品を開発したプロセスを評価するために、製品開発プロセス、分析設計手法、テストと文章化技法、変更管理方針などを調査する。組織が所有しているソフトウェア開発環境、チーム、コンピュータハードウェア、ソフトウェアツールからなる資源が評価対象になる。

ＣＭＭのソフトウェアの品質を対象にしている対し、ＣＭＭＩはシステム全体の品質を対象にしている。成熟度レベルと能力レベルという２種類の評価方法がある。成熟度レベルは組織全体の評価を行うのが目的であり、能力レベルは各プロセスエリア単位の活動を評価し改善するのが目的である。従って、ＣＭＭＩはソフトウェア組織およびプロジェクトのプロセスの成熟度を評価するモデルである。求める答はアとなる。

例題演習

進捗管理に用いられるガントチャートの特徴として，適切なものはどれか。

ア　作業遅れによるほかの作業への影響を明確にすることができる。
イ　作業の順序関係を明示することができる。
ウ　進捗管理上のポイントであるクリティカルパスを明確にすることができる。
エ　日程について予定と実績を対比することができる。

解答解説

ガントチャートに関する問題である。

ガントチャートは、進捗管理用の図表で、横軸に日付、縦軸に工程をとり、計画と実績を横線で記入する。これによって、生産や工事などが計画通りに進んでいるかどうかが明らかになる。この方式は単純でありわかりやすいが、情報量が少ないのが欠点である。

ア、イ、ウの項目はＰＥＲＴ図では表すことができるが、ガントチャートでは十分に表現することができない。求める答えはエとなる。

例題演習

50項目の作業を４人の要員で10日間で完了する計画を立てた。現在５日目を終わった時点で完了したのは20項目である。進捗の遅れを，現在完了した作業項目が本来終わっていなければならない日との差で表すとすると，遅れは何日か。ここで，投入要員の数は変わらないものとする。

ア　１　　イ　２　　ウ　３　　エ　４

解答解説

プロジェクトの進捗管理に関する問題である。

当初計画は50項目を４人で10日間であるから、１人の１日の割合は1.25項目である。

計画では５日で完了する項目数は

$$1.25 \times 4 \times 5 = 25$$

項目である。現在の消化項目数は20項目であるから、

25－20＝５

で５項目不足であり、４人の仕事の日数に換算すると

1.25×４＝５

１日分となる。求める答えはアとなる。

例題演習

あるシステムを開発するための工数を見積もったところ150人月であった。現在までの投入工数は60人月で，出来高は全体の３割であり，進捗に遅れが生じている。今後も同じ生産性が続くと想定したとき，このシステムの開発を完了させるためには何人月の工数が超過するか。

ア　50　　イ　90　　ウ　105　　エ　140

解答解説

進捗管理に関する問題である。

現在までの出来高は0.3で、計画の工数では150×0.3＝45であるが、実際に要した工数は60で、15工数多くかかっている。従って、全体の仕事を完成させるのに必要な超過工数は

15×(１／0.3)＝50

超過工数は50人月となる。求める答えはアとなる。

例題演習

開発期間10か月，開発工数200人月のプロジェクトを計画する。次の配分表を前提とすると，ピーク時の要員は何人となるか。ここで，各工程の開始から終了までの人数は変わらないものとする。

工程名／項目	要件定義	設計	開発・テスト	システムテスト
工数配分	16%	33%	42%	9%
期間配分	20%	30%	40%	10%

ア　18　　イ　20　　ウ　21　　エ　22

解答解説

開発管理に関する問題である。

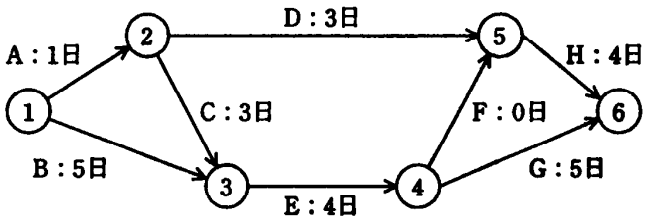
	要件定義	設計	開発・テスト	システムテスト
工数	32	66	84	18
期間	2	3	4	1
人数	16	22	21	18

表の工数配分、期間配分の比率に従って、各工程の工数、期間、人数を求めると、表のようになる。ピーク時の要員は22人となる。求める答えはエとなる。

例題演習

プロジェクトのスケジュール管理のために次のアローダイアグラムを作成した。クリティカルパスはどれか。

ア　A→C→E→G
イ　A→D→H
ウ　B→E→F→H
エ　B→E→G



解答解説

ＰＥＲＴ図からクリティカルパスを求める問題である。

	①	②	③	④	⑤	⑥
ＴＥ	0	1	5	9	9	１４
ＴＬ	0	2	5	9	１０	１４

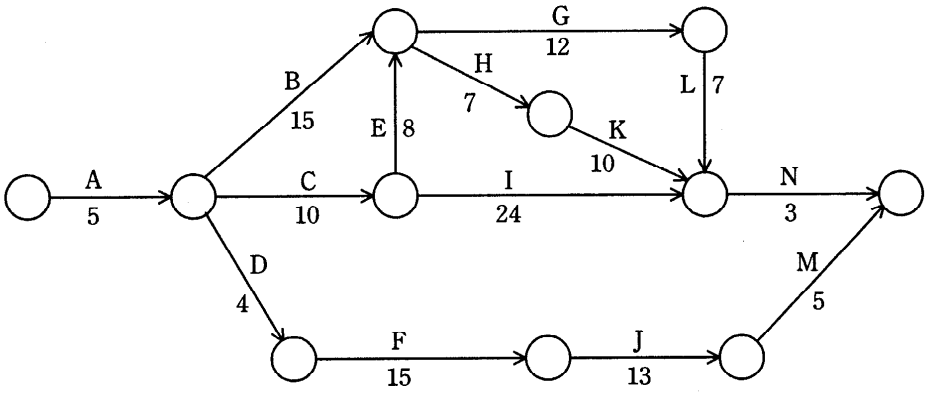
クリティカルパスの求め方は、最早開始時刻（ＴＥ）、最遅完了時刻（ＴＬ）の表を作成し、両者の値が一致するイベントを結合すればよい。

最早開始時刻、最遅完了時刻の表を作成する。

ＴＥ、ＴＬの値が等しいイベントを結合すると、①→③→④→⑥となり、Ｂ→Ｅ→Ｇで、求める答えはエとなる。

例題演習

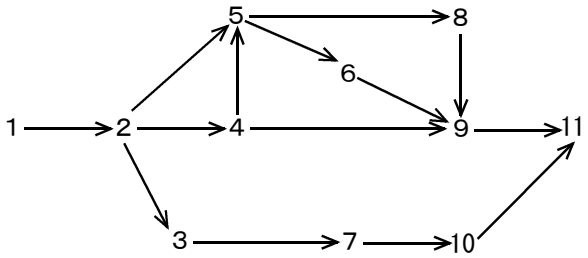
図に示すアローダイアグラムは，あるシステムの開発作業を表したものである。クリティカルパスはどれか。ここで，矢印に示す数字は各作業の所要日数を表す。



ア　Ａ－Ｂ－Ｇ－Ｌ－Ｎ　　イ　Ａ－Ｂ－Ｈ－Ｋ－Ｎ
ウ　Ａ－Ｃ－Ｅ－Ｇ－Ｌ－Ｎ　　エ　Ａ－Ｃ－Ｉ－Ｎ

解答解説

クリティカルパスに関する問題である。



最早開始時刻、最遅完了時刻を求めると次のようになる。

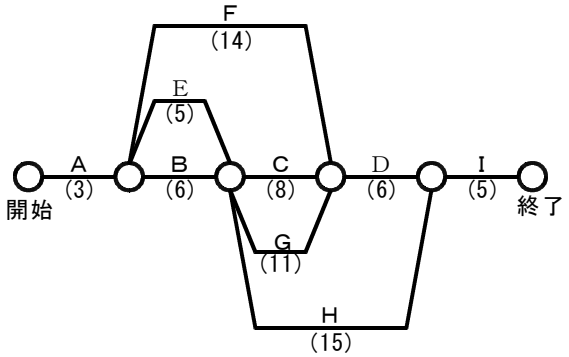
表から、クリティカルパスは、１→２→４→５→８→９→１１となる。Ａ→Ｃ→Ｅ→Ｇ→Ｌ→Ｎとなり、求める答えはウとなる。

	1	2	3	4	5	6	7	8	9	10	11
ＴＥ	0	5	9	15	23	30	24	35	42	37	45
ＴＬ	0	5	12	15	23	32	27	35	42	40	45

例題演習

図は、あるプロジェクトの作業工程（Ａ～Ｉ）とその作業日数を表している。このプロジェクトが終了するまでに必要な最小の日数は幾らか。

ア　２７
イ　２８
ウ　３１
エ　３２



解答解説

パート図のクリティカルパスを求める問題である。

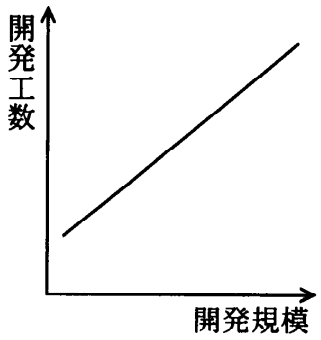
イベント番号	1	2	3	4	5	6
最早開始日	0	3	9	20	26	31
最遅完了日	0	3	9	20	26	31

開始から完了までの最短の経路を求めるとよい。だだし、２点の経路が２通り以上ある場合は最長の経路が求める経路になる。クリティカルパスは、Ａ→Ｂ→Ｇ→Ｄ→Ｉの作業工程で所要日数は31日となり、求める答えはウとなる。

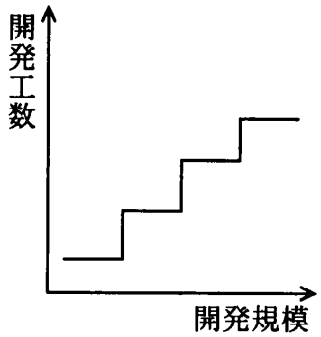
例題演習

ソフトウェアの開発規模と開発工数の関係をグラフで表現したとき，最も適切なものはどれか。

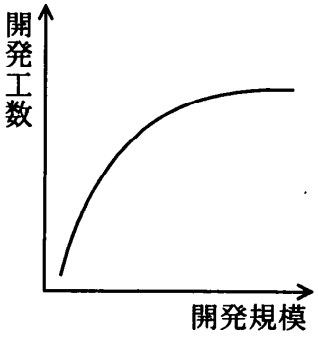
ア



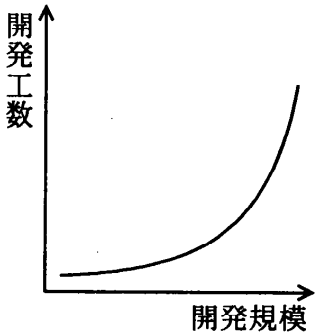
イ



ウ



エ



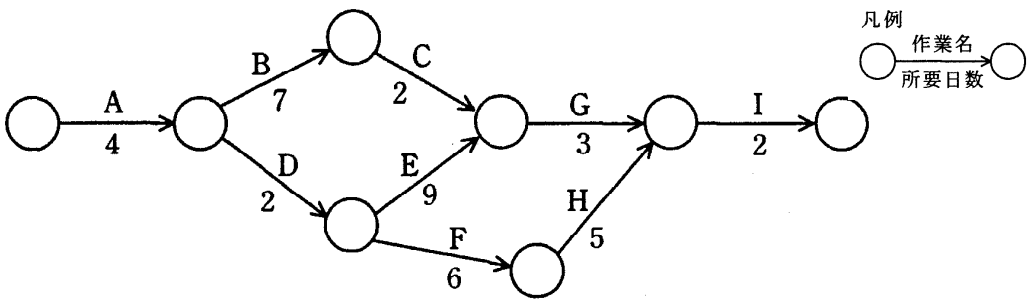
解答解説

開発規模と開発工数の関係に関する問題である。

開発規模の増大と共に開発工数は指数関数的に増大する。求める答えはエとなる。

例題演習

九つの作業からなるプロジェクトがある。作業Ｅの所要日数を９日から６日に短縮すると，このプロジェクトの最短作業日数を何日短縮できるか。



ア　０（短縮できない）　　イ　１

ウ　２　　エ　３

解答解説

パート図に関する問題である。

パート図の最早開始日Ｅの所要日数を９日の場合と６日の場合を求めると、次のようになる。

	1	2	3	4	5	6	7	8
９日	０	４	１１	６	１５	１２	１８	２０
６日	０	４	１１	６	１３	１２	１７	１９

Ｅの所要日数が９日の場合は２０日間、６日の場合は１９日となり、最短作業日数は１日短縮することになる。求める答えはイとなる。

例題演習

ＰＥＲＴを用いてシステム開発プロジェクトの実施計画を作成し，クリティカルパスを算出した。クリティカルパスによって把握できるものとして，適切なものはどれか。

ア　システムの品質上，最も注意すべき作業を把握することができる。
イ　実施順序の変更が可能な作業を把握することができる。
ウ　プロジェクト全体の遅れに直結する作業を把握することができる。
エ　最も費用のかかる作業を把握することができる。

解答解説

ＰＥＲＴ図のクリティカルパスに関する問題である。

クリティカルパスは、プロジェクト全体の完了時期に重大な影響を与える作業工程の集まりで、プロジェクト全体の完了時期を遅らせないためには、クリティカルパスとなる作業を重点的に管理する必要がある。全体の作業時間を短縮したい場合は、このクリティカルパスを短縮することに注力する。クリティカルパスは全体の遅れに直結する作業を把握することができる。求める答えはウとなる。

例題演習

ある新規システムの開発規模を見積もったところ，500ファンクションポイント（ＦＰ）であった。このシステムを構築するプロジェクトには，開発工数の他にシステムの導入や開発者教育の工数が10人月必要である。また，プロジェクト管理に，開発と導入・教育を合わせた工数の10%を要する。このプロジェクトに要する全工数は何人月か。ここで，開発の生産性は１人月当たり10ＦＰとする。

ア　51　　イ　60　　ウ　65　　エ　66

解答解説

開発工数の見積に関する問題である。

開発規模は500ＦＰであり、生産性は10ＦＰであるから、工数に換算すると

500／10＝50(工数)

導入、開発者教育の工数10人月を加算すると　50＋10＝60

プロジェクト管理の工数を含めると　60×1.1＝66(工数)

求める答えはエとなる。

例題演習

システム開発の見積方法の一つであるファンクションポイント法の説明として，適切なものはどれか。

ア　開発規模が分かっていることを前提として，工数と工期を見積もる方法である。ビジネス分野に限らず，全分野に適用可能である。

イ　過去に経験した類似のシステムについてのデータを基にして，システムの相違点を調べ，同じ部分については過去のデータを使い，異なった部分は経験から規模と工数を見積もる方法である。

ウ　システムの機能を入出力データ数やファイル数などによって定量的に評価し，複雑さとアプリケーションの特性による調整を行って，システム規模を見積もる方法である。

エ　単位作業量の基準値を決めておき，作業項目を単位作業項目まで分解し，その積算で全体の作業量を見積もる方法である。

解答解説

ファンクションポイント法に関する問題である。

ファンクションポイント法は､評価対象の領域を明確にし､システムの提供する５つの機能(外部入力・外部出力・外部照会・内部論理ファイル・外部インタフェース)を抽出し、それぞれのデータタイプに対して、単純レベル、多少複雑レベル、複雑レベルの重み付けを加味して点数をつけ計量化する方法である。ファンクションポイントＦＰは次の式で計算する。

ＦＰ＝Ｃ×(Ｐ１＋Ｐ２＋Ｐ３＋Ｐ４＋Ｐ５)

画面や帳票等の入出力インタフェースの多い場合に適している。外部入出力や内部論理ファイル、照会、インタフェースなどの個数や特性などから開発規模を見積もる。

アはＣＯＣＯＭＯのモデル、イは類似法、ウはファンクションポイント法、エはＷＢＳを用いる方法である。求める答えはウとなる。

例題演習

ファンクションポイント法の説明はどれか。

ア　開発するプログラムごとのステップ数を積算し，開発規模を見積もる。

イ　開発プロジェクトで必要な作業のＷＢＳを作成し，各作業の工数を見積もる。

ウ　外部入出力や内部論理ファイル，照会，インタフェースなどの個数や特性などから開発規模を見積もる。

エ　過去の類似例を探し，その実績と差異などを分析評価して開発規模を見積もる。

解答解説

ファンクションポイント法に関する問題である。

ファンクションポイント法は､評価対象の領域を明確にし､システムの提供する５つの機能(外部入力・外部出力・外部照会・内部論理ファイル・外部インタフェース)を抽出し、それぞれのデータタイプに対して、単純レベル、多少複雑レベル、複雑レベルの重み付けを加味して点数をつけ計量化する方法である。ファンクションポイントＦＰは次の式で計算する。

ＦＰ＝Ｃ×(Ｐ１＋Ｐ２＋Ｐ３＋Ｐ４＋Ｐ５)

画面や帳票等の入出力インタフェースの多い場合に適している。外部入出力や内部論理ファイル、照会、インタフェースなどの個数や特性などから開発規模を見積もる。

アはＬＯＣ法、イはＷＢＳ法、ウはファンクションポイント法、エは類似法である。求める答えはウとなる。

例題演習

ＣＯＣＯＭＯモデルにおいて、ソフトウェア開発の生産性に影響を与える要因のうち、影響度の最も大きいものはどれか。

ア　個人／チームの能力　　イ　実行性能上の制約
ウ　製品の複雑さ　　エ　要求される信頼性

解答解説

ＣＯＣＯＭＯのモデルに関する問題である。

ＣＯＣＯＭＯのモデルは、プログラマの作業量を統計的なモデルによって計算式で算定する方法である。中、大規模のシステム開発に適用できるモデルある。ソフトウェアプロジェクトを３種類に分類して名目工数をを見積もり、これに、製品、計算機、人的、プロジェクト的属性による乗数を掛けて最終値を得る。特に、個人／チームの能力が問題になる。

生産性に影響を与える要因は個人／チームの能力であり。求める答えはアとなる。

例題演習

入力，出力などを基に複雑さを加味してシステム規模を見積もる方法であり，開発工数の見積りにも使われるものはどれか。

ア　ＣＯＣＯＭＯ　　イ　標準タスク法
ウ　ファンクションポイント法　　エ　プットナム(Putnum)モデル

解答解説

開発工数の見積法に関する問題である。

アのＣＯＣＯＭＯのモデルは、プログラマの作業量を統計的なモデルによって計算式で算定する方法で、ソフトウェアプロジェクトを３種類に分類して名目工数を見積もり、これに、製品、計算機、人的、プロジェクト的属性による乗数を掛けて最終値を得る。

イの標準タスク法は、実施する作業単位に工数を積み上げていく見積もりの方法である。Ｗ

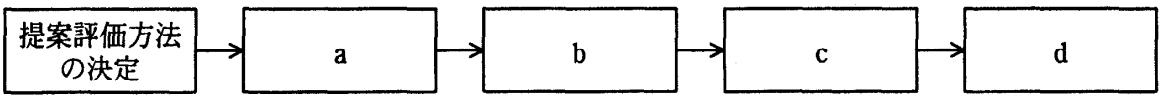
ＢＳ法などを用いて作業単位を設定し、あらかじめ設定してある標準的な工数を各作業に割り当ててボトムアップ的に見積もりを行う。

ウのFunction Point法は、ソフトウェアに含まれる機能単位に見積もる方法で、システムの提供する機能を分類して重み付け係数を掛け、それに対象システムの性格による調整用の特性係数を掛けて最終値を得る。画面や帳票等の入出力インタフェースの多い場合に適している。評価対象を５つの機能に分け、単純レベル、多少複雑レベル、複雑レベルの重み付けを加味して点数をつけ計量化する。求める答えはウとなる。

エのPutnamは、仮説から出発してプロジェクトの工数を表現するモデルで、超大規模システムの開発に適用できるモデルである。

例題演習

“提案評価方法の決定”に始まる調達プロセスを，調達先の選定，調達の実施，提案依頼書（ＲＦＰ）の発行，提案評価に分類して順番に並べたとき，ｃに入るものはどれか。



ア　調達先の選定　　イ　調達の実施

ウ　提案依頼書（ＲＦＰ）の発行　　エ　提案評価

解答解説

調達プロセスに関する問題である。

調達プロセスの次の手順で実施する。

①　提案評価方法の決定

②　提案依頼書の発行（ａ）

③　提案評価（ｂ）

④　調達先の選定（ｃ）

⑤　調達の実施（ｄ）

ｃに該当するのは調達先の選定であり、求める答えはアとなる。

例題演習

情報システムの調達の際に作成されるＲＦＩの説明はどれか。

ア　システム化の目的や業務内容などを示し，ベンダに情報の提供を依頼すること

イ　調達対象システムや調達条件などを示し，ベンダに提案書の提出を依頼すること

ウ　発注元から調達先に対して，契約内容で取り決めた内容の変更を依頼すること

エ　発注元と調達先の役割分担などを確認し，契約の締結を依頼すること

解答解説

ＲＦＩに関する問題である。

ＲＦＩは、企業が調達や業務委託を行う際、自社の要求を取りまとめるための基礎資料として、外部業者に情報の提供を要請すること。あるいはその要請をまとめた文書をいう。

ＲＦＩの目的は、知りたい情報を文書として明確化することで、明快な回答を確実に得ることである。複数業者の回答を比較する場合も同一の質問に対する答えであれば、検討しやすい。ＲＦＩの発行は、提案や見積もりといった交渉の前段階で行われるもので、数多くの外部事業者から、自社が求める能力を持ち、交渉に足る相手を絞り込むといったことも狙いとなる。ハイテクを対象としたＲＦＩでは、自社が知らない新製品・新技術に関する知見を得ることも目的となる。ＩＴ調達におけるＲＦＩはＩＴベンダの技術要件を確認するもので、どのような技術を保有しているか、どのような経験があるかなどを確認するものとなる。

システム化の目的や業務内容などを示し、ベンダに情報の提供を依頼することである。求める答えはアとなる。

アはＲＦＩ、イはＲＦＰ、ウはＲＦＣ、エはＳＯＷである。

例題演習

契約タイプで一括請負契約に属するものはどれか。

ア　請け負った作業の履行に対するコストが償還され更にプロジェクトのコスト見積りに対して一定比率の固定フィーを受け取る。

イ　請け負った作業の履行に対するコストが償還され事前に取り決めたフィーと，契約で定めたパフォーマンス目標レベルの達成度に応じたインセンティブを受け取る。

ウ　契約で合意した内容を実現するために，実施された労務に対する対価が支払われる。

エ　契約で合意した内容を実現するために，指定された期日までに決められた価格で作成された成果物に対して対価が支払われる。

解答解説

請負契約に関する問題である。

請負契約は受託者が一定の業務を完成し、その結果に対して発注者が対価を支払う契約形態である。受託者は成果物の完成責任がある。また、成果物に対して瑕疵担保責任がある。システム開発要員の指揮命令は受託者が行う。受託者が下請けを使用することも可能である。

ア、イの請け負った作業に対するコスト、一定比率の固定フィーの考え方では請負の場合は契約しない。成果物に対して対価を支払う。

ウの実施された労務に対する対価の支払いではない。

エの期日までに決められた価格で作成された成果物に対する対価が適切な契約形態である。求める答えはエとなる。

txt040413 システムの移行・運用管理

① システムの導入

ⓐ システムの導入準備

システムやソフトウェアを導入する際に必要な準備作業には次のものがある。

㋐ ソフトウェアのインストール
㋑ 受入テスト
㋒ システムの移行に伴う業務手続きの変更
㋓ 利用者と運用担当者への教育
㋔ データの変更作業

ソフトウェアを導入する場合、必要な準備作業を洗い出し、スケジュール、体制、責任者を明確にし、導入計画によって導入作業を効率的に正確に進める。

ⓑ 受入テスト

受入テストは納品・導入されたソフトウェアやシステムが、ユーザや発注者の要求通りに機能を実装していることをユーザや受注者が確認するテストである。テスト内容はシステムテストや運用テストとほぼ同じであるが、検収を意識したテストである。開発者側からエンドユーザにシステムを引き渡す場合、エンドユーザ独自でテストデータを作成して実施し管理する。

② システムの移行

ⓐ システムの移行とは

システムの移行は、新システムを稼働させるために、現行システムから新システムへ、ハードウェアやソフトウェア、各種ファイルを円滑に移し変えることである。そのために、移行方法や移行手順、移行体制、移行日程計画、移行タイムチャートなどを作成する。

ⓑ システム移行のための作業

移行作業の中で大きな比重を占めるものがデータの移行である。利用部門と協同して、現行の業務フロー、データベース仕様を元に、現行システムと新システムのデータ項目を比べ、移行対象データ項目を決める。

ハードウェア、ソフトウェア、データの移行方法、移行のタイミングも重要な検討項目である。新たなシステム開発に伴って、ハードウェアの入れ替えや増強、ＯＳやミドルウェアの入れ替えや変更、通信回線の新設や増設が行われる。

ⓒ　移行方式の種類

㋐　一括移行方法

新システムへの切り替えを、一時期に全面的に行う方式である。移行手順はシンプルであるが、一時期にすべてを移行するのでリスクが大きく、新システムの信頼度が要求される。システムテスト、運用テストにおいて実環境におけるテストを行い、移行にあたっては、移行リハーサルを行うなどして、十分な検証を行う必要がある。

㋑　順次移行方式

特定の地域を対象に試行導入を行い、一定期間の検証後、順次、全面的にあるいは数ステップに分けて展開を行う方式である。移行時のトラブルは局所化できるメリットがあるが、新旧両システムを同時に運用するため、作業負荷が大きくなる。

㋒　サブシステム順次移行

システムを、いくつかのサブシステムに分けて順次開発し、移行していく方式である。一次開発、二次開発、…、に分けて開発を進める。移行時のトラブルはサブシステム内に限定され、移行負荷も平準化されるメリットがあるが、一次開発、二次開発に分割するサブシステム間のインタフェースに関係する課題の検討が重要になる。

㋓　並行運用方式

並行運用方式は、新旧システムを並行稼働しながら移行する方式である。

③　運用管理

ⓐ　運用管理とは

運用管理はシステムを安定的に稼働させるために行う一連の業務である。主な業務内容として次のものがある。

㋐　稼働・休止スケジュール管理
㋑　データのバックアップ
㋒　ハードウェアおよびソフトウェアの資源管理
㋓　ハードディスクやネットワークなどの資源管理
㋔　消耗品管理
㋕　貸し出し用の機器や代替機器の備品管理
㋖　操作マニュアルなどの文書管理
㋗　セキュリティ管理業務
㋘　コスト管理業務
㋙　窓口業務

ⓑ　コスト管理

コスト管理はシステム開発やシステムの運用で必要となる費用を予算として算出し、進捗に応じて実際にかかった費用を対比し収支を管理することであり、システムの運用管理の一つである。管理手順は次のようになる。

㋐　システム概要をもとにした概算コストの見積もり
㋑　作業工程数や作業難易度が明確になった時点での詳細コストの見積もり
㋒　実際の費用管理方法の決定
㋓　実績の集計
㋔　予算と実績の比較分析

ⓒ　運用コスト

運用コストには次のものがある。

㋐　ハードウェアとソフトウェアのリース料や償却費
㋑　ソフトウェア使用料
㋒　通信回線使用料
㋓　保守費用

ⓓ　ＴＣＯ、ＮＣ、ＮｅｔＰＣ

㋐　ＴＣＯ

ＴＣＯは、コンピュータシステムを導入・運営するためにかかる総費用のことである。次のものを含めたコストの総計である。

❶　ハードウェアやソフトウェアの購入費
❷　ソフトウェアのアップグレード費用
❸　エンドユーザの教育費
❹　管理運営費

㋑　ＮＣ

ＮＣは、オラクル社が提唱したＴＣＯ削減を目的とするコンピュータシステムである。外部記憶装置を持たず、データやソフトウェアはサーバからダウンロードし、編集・作成したデータもサーバに保存することによって、運用管理コストを削減する。

㋒　ＮｅｔＰＣ

ＮｅｔＰＣは、マイクロソフトとインテルが発表したＴＣＯの削減を目指した構想である。データやソフトウェアの管理をサーバを使用して実現する。

④　資源管理

ⓐ　ハードウェアの資源管理

ハードウェア資源管理は、コンピュータやその周辺機器を管理することで、管理者は各種ハードウェア機器の利用状況を把握し、資源が有効活用されているかを確認する。

管理の基本的な考え方として、各機器の負荷が分散し、稼働率が平均的に適切な水準を維持して、有効に活用される状態を確保する。更に、耐用年数を考慮して、ある一定の期間を経過した機器は障害発生が多くなるため、機器の入れ替えを検討する。ハードウェア資源を管理するには、システムや機器のレスポンス性能や処理能力などの利用状況を把握するためのデータを収集し、定期的に分析・評価する。

ⓑ　ライブラリの管理

ライブラリ管理は、システムの性格によって異なる。高速処理を要求されるシステムやセキュリティを要求されるシステム、無停止を要求されるシステムなど各種のシステムごとに管理の考え方が異なる。一般的には、運用仕様書や標準運用手順書で定められた基準や規定に従って管理する。

ⓒ　ライブラリ管理業務の内容

㋐　データやプログラムの処理格納場所の明確化
㋑　バックアップライブラリの所在の明確化
㋒　バージョン管理
㋓　機密保護
㋔　汚染保護
㋕　媒体物や入出力物の入出庫管理
㋖　ライブラリの登録、変更、削除管理
㋗　著作権、意匠登録保護、プライバシ保護

⑤　構成管理・変更管理

ⓐ　構成管理

構成管理は、情報システムを構成するハードウェア、ソフトウェア、ネットワークなどの構成要素を管理台帳などに記録して管理することである。システムの構成要素は環境の変化に伴い、随時変更される。システム構成要素の変更時にその変更内容をタイムリに記録したり、変更履歴を管理することによって、安定的なシステム運用を維持する。

ⓑ　ソフトウェアの構成変更管理

構成要素であるソフトウェア品目の基準からの変更依頼、修正内容、修正が必要となった場合の関係部署への通知、修正されたソフトウェアの版、リリース版などの管理を行う。構成変更管理を行うことにより、システムの構成要素が変更された場合の追跡性、完全性を図る。

ⓒ　バージョン管理

バージョン管理は、ソフトウェアや利用マニュアルのバージョンを管理することである。ソフトウェアやマニュアルは、バージョンによって機能や内容が異なるため、現在使用中のバージョンを正しく認識しておく必要がある。システムに不具合に対処するには、バージョン管理が決め手になる。

ⓓ　バージョンアップ

バージョンアップは、ハードウェアやソフトウェアの機能追加や改良を施して、新しいバージョンにすることである。旧バージョンの製品で作成したファイルが新バージョンの製品でも使えるように下位互換性を保証することが一般的である。

バージョンアップを実施する場合、ＯＳやデータベースシステム、業務アプリケーションなど業務に使用している一連のソフトウェアが新しい環境で稼働可能なことを確認してから、問題がなければバージョンアップすることになる。

ⓔ　ソフトウェアの廃棄

廃棄はソフトウェアライフサイクルの最終段階のプロセスであり、次のような場合に、ソフトウェアは廃棄の対象になる。

㋐　機種の変更などによって継続利用が不可能または困難になったソフトウェア
㋑　利用者から継続使用の申請がないソフトウェア
㋒　利用頻度の少ないソフトウェア

⑥　データ資源の管理

ⓐ　データ管理の部門別役割

ユーザの役割はデータそのものの入出力管理や信憑性に責任を持ち、運用部門はユーザの指示のもとで、正常に、安定的に、決められたサービスレベルの範囲で利用可能にする責任を持つ。データ管理者は、開発面では業務の実世界から概念設計を行い、システム化の範囲で論理設計を行い、運用面では組織全体の管理や調整を行い、データベース管理者は、論理データモデルから物理設計を行い、データベースを構築、構築後のデータベースの運用設計および運用保守を担当し、個別業務単位に技術的な面から管理を行う。

ⓑ　データ資源管理の業務内容

㋐　信頼性の確保
㋑　不正使用防止管理
㋒　機密保護
㋓　データ取り扱い方法の体系化、標準化、規則化
㋔　データの監査

ⓒ　データの廃棄

データの利用に際しては、効率的な利用方法とセキュリティ保持が重要である。廃棄後のデータは管理されないため、重要情報が漏洩しやすい。重要情報には、売り上げ、利益、財務状況、事業計画に関する情報、個人の秘密に関する情報などがある。不要になったデータの廃棄に当たっては、重要データの漏洩を防止するため、厳重なチェックが不可欠である。データの廃棄に当たっては、適切な方法の選択の他にも管理上、留意すべきことがある。特に、セキュリティ上の必要性とデータ保全の必要性を考慮することが重要である。

ⓓ　データ廃棄の方法

廃棄手段	内容、特徴、留意点
消磁	磁気ディスクや磁気テープに保存された磁気データを消してから廃棄する 情報システムにおけるデータ廃棄の主要な方法である。
破壊	磁気媒体以外に保存されたデータの廃棄の際に用いる。 焼却が困難な媒体を使用している場合に有効である。
焼却、溶解	紙の上に記録されたデータを廃棄するのに最も適した方法である。 廃棄量が多くなるため、外部の専門業者に委託することが行われる。 セキュリティ上の問題が発生する恐れがあるため、書類の内容が見えない状態で外部に出す配慮が必要になる。
裁断	機密性の高い書類データを廃棄する際に用いられる方法である。 焼却と溶解の組合せが考えられる。

ⓔ　データ廃棄に関する管理上の留意点

㋐　廃棄記録簿の作成

記録簿には、データの名称、保存形態、廃棄の年月日、廃棄の方法、廃棄の担当者と責任者などのデータを記録しておく。記録簿を作成することによってセキュリティなどの管理精度が向上する。データの保存状況を表す情報と併せて使用することにより廃棄計画も立てやすくなる

㋑　立会者、確認者の義務づけ

重要データが漏洩しないようにした上で、データの廃棄を行うためには、作業上の統制が確立していなければならない。作業担当者以外にチェックのための立会者、確認者をおく。廃棄してはならないデータが誤って、あるいは故意に廃棄されることを防止するための確認体制を確立する。

㋒　廃棄業者との契約

廃棄業者を利用する場合は廃棄方法や機密保持、下請業者の利用禁止などを盛り込んだ契約書を取り交わす。同時に損害賠償責任についても明らかにしておく。

⑦　データの保全

ⓐ　バックアップ作業

大規模情報システムでは、データが消失すれば復旧作業は膨大になり、復元の精度も期待できない。特に、磁気的なデータは誤動作や誤操作で消失しやすいので、必要に応じてデータやプログラムの複製を作成するバックアップ作業が不可欠になる。障害時にデータの速やかな回復を行うためには、定期的なバックアップ作業が必要である。バックアップ作業には、大量のハードウェアと長時間の作業が必要であり、通常の業務遂行の障害になる。従って、バックアップの方法や間隔はデータの重要度に応じて決定する必要がある。

ⓑ　バックアップの方法

㋐　世代管理のバックアップ

一定周期で作成されるデータを作成される度に保存していく方法である。

㋑　ファイルの多重化

同一の構造、同一の内容をもつデータを常に２組以上用意しておき、同時に更新する方法である。維持のためのコストは高くなるが、オンラインシステムには有効である。

㋒　バックアップコピーの採取

データを一定間隔でコピーし、別のファイルとして保管する。障害時には、バックアップコピーを利用して回復させる。バックアップコピーとログ情報を利用して復旧させる。

㋓　遠隔地保管

処理システムと離れたところにデータを保管し、通信回線を利用して回復処理を図る。多重化して分散配置が可能で、局所的なトラブルに強い方法である。

ⓒ　大規模データベースのバックアップ

大規模なデータベースのバックアップはバックアップ時間を考慮して、週次ではフルバックアップ、日次では差分バックアップを活用して効率的に行う。

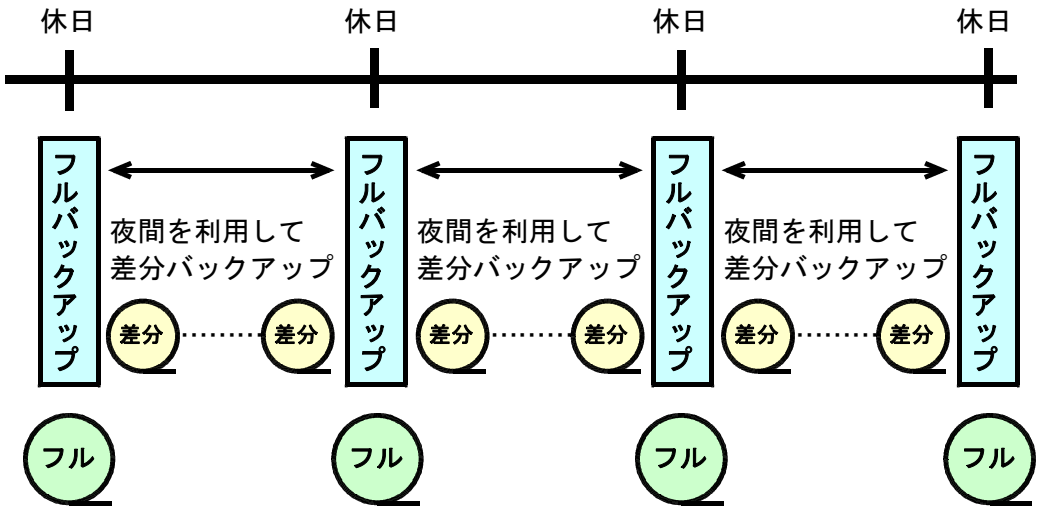
ⓓ　フルバックアップと差分バックアップ

㋐　フルバックアップ

フルバックアップはバックアップ対象のすべてのプログラムやデータを外部の媒体にコピーする方式である。データ採取時に、大量のハードウェアと長時間の作業が必要となる。バックアップ採取の頻度とタイミングが問題となる。

㋑　差分バックアップ

差分バックアップはデータの更新部分だけをバックアップする方式で、前回のフルバックアップ採取時以降、更新のあった部分だけのコピーを採取する。フルバックアップに比べて情報量が少ないために採取時間が短く、必要な媒体の量も少なくてすむ。障害回復時には、フルバックアップと差分バックアップの両方を利用してロールフォワードを行うため、回復作業が複雑で時間が長くなりやすい。通常は、回復処理の実行頻度に比べ、バックアップの頻度はかなり大きいので、差分バックアップ機能の利用は有効である。



ⓔ　差分バックアップ作業の特徴

㋐　復旧のためには差分バックアップを採取する直前までのフルバックアップとその後の差分バックアップを組み合わせて可能となる。

㋑　差分バックアップを行うためにはフルバックアップが必要であり、ある期間のバックアップ処理時間短縮のためには交互運用の可能性はある。

㋒　差分バックアップを用いてデータベースのロールフォワードを行う場合、フルバックアップとそれを採取後の差分バックアップの全てを利用しなければならないので回復操作が複雑

になり回復時間が長くなる。

㊓　差分バックアップの処理時間はフルバックアップの処理時間に比べて情報量が少ないだけ処理時間が短くなる。

⑧　システムの保守作業

ⓐ　保守作業

システムの本番稼働後に、システムに潜んでいたバグが発見されたり、ユーザによるシステムの仕様の変更依頼があった場合、システムを修正しなければならないケースが生じる。この場合の作業を保守という。保守作業では、プログラムの修正、追加などを行った後、文書を修正しておく必要がある。稼働しているシステムの処理内容と文書の内容が異なると、不都合が生じる。従って、書式を定め、それに内容を書き込み、責任者の承認を取り、文書の修正も同時に行える手続きの標準化が必要である。

ⓑ　保守作業の分類

㋐　修正作業

プログラムの誤りや、処理・手続きなどの仕様の誤りであるバグによる不具合の修正作業である。優先順位が最も高く、早急に対処する必要がある。

㋑　変更作業

法律の改正や金利の変更、年号の変更などの外部の要因による変化への対応やデータの変化などの対応作業になる。修正作業と共に最優先の作業である。

㋒　改良作業

ユーザ部門からの処理追加や機能追加の要求、不十分な評価をされた処理機能の改良に対処する作業である。最も工数がかかる作業になる。

⑨　保守体制

ⓐ　保守体制とは

最適な保守を行うためには、保守体制を組織する必要がある。保守体制には、次の職能が必要となる。

㋐　保守管理者

㋑　変更管理者

㋒ **保守担当者**

㋓ **構成管理者**

ⓑ　保守体制の考え方

㋐ **高度な能力が要求される。**

㋑ **保守専任体制とする。**

㋒ **保守担当者は、システム開発の段階から参加する。**

㋓ **データや文書、ライブラリを管理する担当者を任命する。**

㋔ **ユーザ対応の責任者を任命する。**

ⓒ　各管理者、担当者の職務内容

㋐　保守管理者

保守に関してユーザ対応の受付窓口の役割を果たす。保守に関するすべての依頼は保守管理者を通して受け付ける。変更管理者と保守の内容について、変更内容、納期、費用対効果など、その妥当性の検討を行い、保守を行うかどうかを決定する。保守を行うことが決まると、保守担当者に作業指示を行う。

㋑　変更管理者

保守管理者と協力して、保守内容の妥当性を検討する。変更内容を文書化し、履歴として管理する。

㋒　保守担当者

保守管理者の指示のもとに、保守作業を実際に行う。作業が完了すると、保守管理者と構成管理者に報告する。

㋓　構成管理者

システムのバージョン管理を行う。使用しているバージョンがユーザによって異なる場合、使い勝手や他のソフトウェアとの連携などから不都合が生じるので、正確に管理する必要がある。

ⓓ　保守手順とその作業内容

㋐ **ユーザは保守管理者に保守を依頼する。**

㋑ **保守管理者は変更管理者やシステムの責任者とその妥当性について検討し判断する。**

㋒ **保守管理者は保守担当者に保守作業の指示を行う。**

㋓ **変更管理者はシステムの変更内容を文書化する。**

㋔ **保守担当者はシステムの変更作業、テストを実施する。**

㋕　保守担当者は進捗状況を保守管理者に報告し、作業完了結果を保守管理者と構成管理者に報告する。

㋖　構成管理者は対象システムのバージョンを更新する。

⑩　保守の形態

ⓐ　予防保守

障害の発生を防ぐための保守であり、あらがじめ保守計画を立てて実施する。保守要員の確保も計画的に行えるため、効率の良い保守が実施できる。予防保守には、日常保守と定期保守がある。日常保守は、システムを構成する機器の状態や性能を監視するために、常日頃実施する。定期保守は、あらかじめ定められた一定期間ごとに実施する。比較的大がかりな作業で、システムを停止するか、代替機などに切り替えて実施する。

ⓑ　事後保守

障害が発生したり、発生する恐れがある場合に実施する保守である。この保守はあらかじめ定められた作業でないために、保守要員の確保が難しい。事後保守には臨時保守と緊急保守がある。臨時保守はシステムの運用を行っている際に、通常の状態と異なる状況が発生した場合、臨時的に実施する。障害が発生する前に対策をとり、障害を未然に防ぐことが目的である。緊急保守はシステムに障害が発生したときに、障害を修復する目的で実施する。機器の切り離しなどで、システムを一次的に停止する。緊急保守件数が増加傾向にある場合は、システム全体の見直しを含めた保守計画を立て、実施する必要がある。

⑪　ソフトウェアの保守

ⓐ　ソフトウェアの保守とは

ソフトウェアの保守は機能の追加や改善、修復、予防などを目的とするものに分類できる。機能の追加や改善は、稼働中のシステムの機能面の要件を変更するためのもので、ソフトウェアの機能仕様変更や性能改善に対して保守が行われる。稼働しているシステムのソフトウェアの修復には、要求性能や要求仕様を満たしていない場合に発生する修正保守、業務要件や処理環境の変化により生じる適応保守、ソフトウェアの完成度を高めるための完全化保守などがある。予防保守は、運用時に予想されるシステム障害や運用問題に対して、トラブルを事前に予防する目的で技術支援や予防修正などを実施する。

ⓑ　システムのライフサイクル

システムの企画に始まり、基本計画から開発、運用、保守、システムの陳腐化に伴って、新しいシステムにリプレースされ、廃棄されるまでの流れをシステムのライフサイクルという。

情報システムは環境の変化に伴う業務処理の変化やデータ量の増大に対して、必要部分に修正を加えて運用を継続する。

ⓒ　システムリプレースの発生原因

㋐　修正の負担が新規開発のコストを上回るようになる場合
㋑　新たな技術や機器を導入することで費用対効果が期待できる場合

⑫　アウトソーシング

ⓐ　アウトソーシングとは

アウトソーシングとは、企業内で一定の範囲を持った業務や部門の企画・設計から運営までを、専門知識やノウハウを持った外部業者に委託することである。システムの高度化、複雑化に伴って、日常の運用および保守に従事する要員に高度な技術力が求められるようになり、その要員の確保と育成が困難になってきた。システムの保守を実施する場合、専任の保守要員が必要になる。しかし、専任の保守要員を配備しておくことはコスト的に問題になる場合がある。保守作業を保守契約により、外部に委託すること(アウトソーシング)を行う。

ⓑ　保守契約により作業を外部に委託する利点

㋐　保守要員を常駐させるよりもコストを低く抑えることができる。
㋑　保守要員を準備する必要がなくなり、要員問題を解決できる。
㋒　社員に24時間体制の勤務をさせる必要がなくなる。
㋓　専門家による詳細な保守が期待できる。

ⓒ　保守契約により作業を外部に委託する欠点

㋐　セキュリティ面で問題が発生する恐れがある。
㋑　委託先人事により保守員が変更される場合もある。
㋒　保守員が優秀な人材とは限らない。
㋓　運用面での問題提起が行われない。

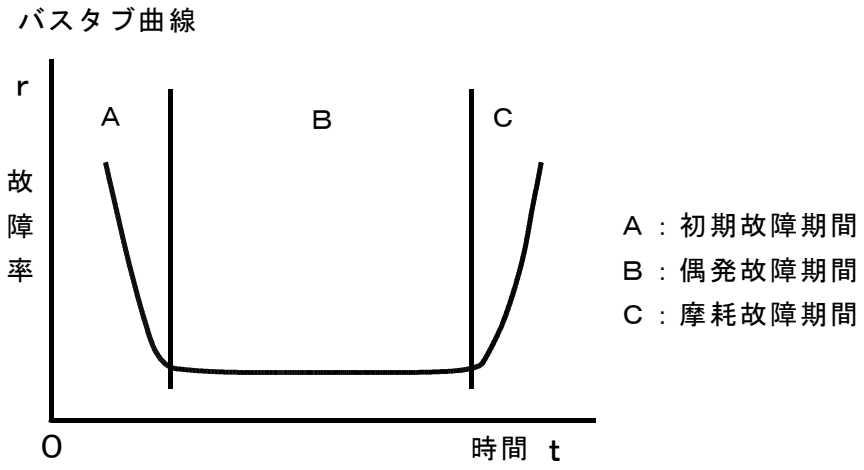
⑬　バスタブ曲線

ⓐ　バスタブ曲線とは

ハードウェアの故障発生頻度の時間的変化を表す曲線で、縦軸に故障率、横軸に時間をとっ

たとき、その曲線の形がバスタブ型になることから呼ばれる。

ⓑ　システムの稼働後の故障発生原因の分類



㋐　初期故障

設計ミス、製作ミスによるもので、時間の経過とともに減少する。

㋑　偶発故障

故障の発生が激減し、安定する。

㋒　摩耗故障

器材の摩耗や劣化による故障で、時間とともに激増する。

時間の経過による故障率の変化は、図に示すようなバスタブ曲線となる。

⑭　障害管理の目的

ⓐ　障害

システムの障害はシステムの応答時間の悪化、帳票出力の誤り、機器の障害、端末表示の誤り、業務処理の全面停止などで、システムが利用できなくなることである。

情報処理システムは、高機能化、複雑化、大規模化しており、比例して障害の種類も増加し、複雑化している。システムの障害に対して、障害の局所化、未然防止および再発防止などを目的とする障害管理はシステム運用部門の重要な業務である。

ⓑ　障害管理の目的

㋐　障害が発生しない、または発生させない情報システムの実現を追求する。

㋑　障害が発生した場合、早期発見、早期対応、障害の局所化を図る。
㋒　原因の究明、再発防止、未然防止の対策を実施する。
㋓　障害対応の手順化、手続化とその実施訓練を徹底する。

ⓒ　障害対策時の基本的な考え方

㋐　関連部門への迅速な連絡
㋑　障害範囲の的確な把握
㋒　障害の拡大防止
㋓　二次障害の発生防止
㋔　原因究明の迅速化
㋕　復旧時間の短縮

ⓓ　障害管理部門の業務

㋐　障害監視
㋑　障害原因の究明
㋒　影響範囲の見極めと局所化
㋓　関連部門への周知と協力要請
㋔　障害回復処理
㋕　障害の記録と報告
㋖　再発防止と障害時間短縮のための改善

⑮　障害管理の作業

ⓐ　事前作業

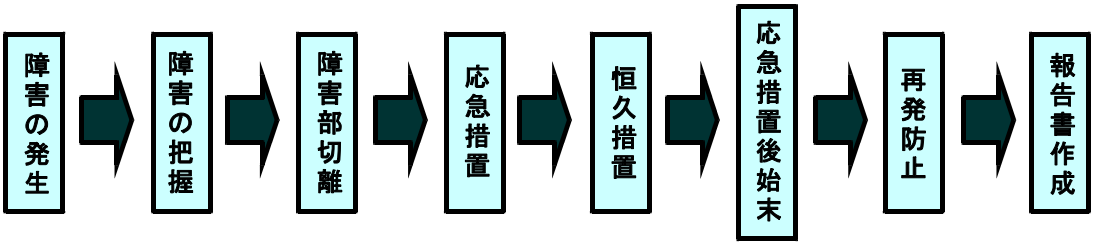
㋐　障害監視項目や監視内容、監視方法の決定
㋑　障害時運用マニュアルの作成、変更
㋒　回復処理で使用する各種データの保存
㋓　障害時に採取するデータ、資料の決定

ⓑ　障害発生時作業

㋐　関連部門への障害発生の連絡
㋑　原因究明のためのデータ、情報収集
㋒　原因の切り分け、究明

㊓　局所化
㊔　回復処理
㊕　障害回復の連絡

ⓒ　障害発生時の一般処理手順



ⓓ　事後作業

㋐　原因判明に至らなかった障害の原因追及、または他の情報処理技術者への追求依頼、進捗状況の管理
㋑　原因判明後の対応および障害発生時に回復処理として、臨時措置をとった場合の復旧処理
㋒　復旧処理後一定の安定期間をみて、不要となったプログラム、パッチ、データなどを処分
㋓　障害の記録、報告
㋔　再発防止および障害時間短縮のための改善策検討、実施

⑯　障害情報の種類

ⓐ　障害の発生を知らせる情報

㋐　システムコンソールに表示されるメッセージ
㋑　端末装置に表示されるメッセージ
㋒　情報処理機器の制御盤上の情報
❶　異常ランプの点灯、異常番号の表示
❷　計器の数値
❸　表示メッセージの内容
❹　警告音の鳴動
㋓　プログラムの実行結果
帳票の出力状況、端末の表示内容

ⓑ　障害の原因を究明するための情報

㋐　ログ情報
㋑　システムソフトウェアのトレース
㋒　ダンプ情報
㋓　障害発生前後のオペレーションの手順・内容
㋔　監視機器の状況

⑰　システムの安全対策

ⓐ　要素の信頼性対策

システムの安全性を高めるには、システムを構成する施設、ハードウェア、ソフトウェア、データ、ネットワークおよび人などの各要素の信頼化対策が重要である。

検討する対策は、災害対策と不正行為防止対策がある。不正行為防止対策の詳細はセキュリティの検討項目として別途取り上げる。

ⓑ　災害対策

災害には、自然災害と人災がある。自然災害には、地震、風水害、雪害、落雷、塩害などがあり、人災には、火災、事故などがある。災害対策は、設備面での対策が主であり、次の点について考慮する。

㋐　建物の構造強化

耐火構造、耐水・防水構造、耐震構造などを検討する。

㋑　防災設備の強化

❶　防火設備：自動火災報知器設備など
❷　防水設備：漏水検知器、止水弁など
❸　設備の耐震化：各機器への耐震装置の設置など
❹　電源設備の防災対策：無停電設備の設置、予備電源設備、電源の２系統化など
❺　回線設備の防災対策：回線の２系統敷設、有線回線と無線回線の併設など

㋒　バックアップセンターの設置

❶　専用型バックアップセンター
❷　相互型バックアップセンター
❸　共用型バックアップセンター

ⓒ　不正行為の防止対策

不正行為には、不法侵入や詐取などによる犯罪行為とコンピュータシステムの特性に乗じて行われるコンピュータ犯罪の２種類に大別できる。

不法侵入による犯罪行為防止対策には、建物、コンピュータ室の構造強化、コンピュータ室への出入りに関するチェックの厳重化、ＩＴＶの設置、記憶媒体の保管場所の二重化、金庫保管の実施などが検討される。

コンピュータ犯罪は、データの漏洩、破壊、改ざんなどシステムの不正使用に関するものであり、認証、アクセス制御、ファイアウォールの設置、隔離、監視の強化などのセキュリティ対策が必要になる。

ⓓ　コンピュータセンターのバックアップの方式

㋐　専用バックアップセンタ

地理的に離れた場所に、専用のバックアップセンタを用意し、定期的にプログラムやデータを送信する。ホットサイトとコールドサイトの方式がある。ホットサイトは遠隔地のシステムに、障害時に直ちに引き継げるようにプログラムやデータの更新情報を絶えず送信しておく方法である。コールドサイトは、切り替えに時間を要する方式である。

㋑　商用バックアップセンタ

商用のバックアップセンタと契約を結び、障害発生時に設備を提供してもらう。

㋒　相互援助

他企業と相互バックアップ援助の契約を結ぶ。

システムの重要性の変化に伴って、施設や設備の設置基準の見直しが必要である。

例題演習

システムの一斉移行方式の特徴に関する記述として，適切なものはどれか。

ア　運用方法はシステム稼働後に段階的に周知されるので，利用者の混乱が避けられる。
イ　システム規模が小さい場合に行われ，移行に失敗した場合の影響範囲を限定できる。
ウ　新旧システムを並行して運用することによる作業の二重負担を避けることができ，経済的効果が大きい。
エ　新システムの処理結果と従来システムの処理結果を比較しながら運用することができ，問題がなければ比較作業を一斉にやめて新システムに移行できる。

解答解説

システムの移行方式に関する問題である。

システムの移行とは、新システムを稼働させるために、現行システムから新システムへ、ハードウェアやソフトウェア、各種ファイルを円滑に移し変えることである。そのために、移行方法や移行手順、移行体制、移行日程計画、移行タイムチャートなどを作成する。

移行方式の種類

一括(一斉)移行方法は、新システムへの切り替えを、一時期に全面的に行う方式である。移行手順はシンプルであり、二重負荷を避け、経済効果も大きいが、一時期にすべてを移行するのでリスクが大きく、新システムの信頼度が要求される。システムテスト、運用テストにおいて実環境におけるテストを行い、移行にあたっては、移行リハーサルを行うなどして、十分な検証を行う必要がある。

ア、エは順次移行方式、イは順次移行方式、サブシステム順次移行方式の考え方、ウは一斉移行方式である。求める答えはウとなる。

例題演習

一斉移行方式の特徴のうち，適切なものはどれか。

ア　新旧システム間を接続するアプリケーションが必要となる。
イ　新旧システムを並行させて運用し，ある時点で新システムに移行する。
ウ　新システムへの移行時のトラブルの影響が大きい。
エ　並行して稼働させるための運用コストが発生する。

解答解説

システムの移行方式に関する問題である。

一括(一斉)移行方法は、新システムへの切り替えを、一時期に全面的に行う方式である。移行手順はシンプルであり、二重負荷を避け、経済効果も大きいが、一時期にすべてを移行するのでリスクが大きく、新システムの信頼度が要求される。システムテスト、運用テストにおいて実環境におけるテストを行い、移行にあたっては、移行リハーサルを行うなどして、十分な検証を行う必要がある。

ア、イ、エの方式は順次移行方式の考え方である。ウは一斉移行方式の問題点を記している。求める答えはウとなる。

例題演習

システムの開発部門と運用部門が別々に組織化されているとき，開発から運用への移行を円滑かつ効果的に進めるための方法のうち，適切なものはどれか。

ア　運用テストの完了後に，開発部門がシステム仕様と運用方法を運用部門に説明する。
イ　運用テストを効率良く行うために，開発部門の支援を受けずに，運用部門だけで実施する。
ウ　運用部門からもシステム開発に積極的に参加し，運用性の観点から助言する。
エ　開発部門は，運用テストを実施して運用マニュアルを作成し，運用部門に引き渡す。

解答解説

システムの移行と運用部門の関係に関する問題である。

アの運用テストの完了後に運用部門に移行する方式では、開発部門が実施する運用テストにも問題が発生し、運用部門への移行が円滑には進まない。

イの運用テストの方法は、運用部門が主体性をもち、開発部門が支援する体制で行うのが最も効果的である。

ウのシステム開発段階から運用部門が積極的に参画し、運用性の観点から支援する記述内容は適切である。求める答えはウとなる。

エの開発部門が運用テストを実施してマニュアルを作成し、運用部門に引き渡す方法では、運用テストの実施、マニュアルの内容、運用段階に問題が発生し効果的な方法にはならない。

例題演習

システムを運用管理の立場から評価する場合、可用性評価の対象となるのはどれか。

ア　オンラインシステムの応答時間が短い。
イ　オンラインシステムの障害復旧が早い。
ウ　オンライン端末の操作が簡単である。
エ　他人のデータを本人の許可なく参照することができない。

解答解説

可用性評価に関する問題である。

可用性はシステムが故障しないで利用したいときにいつでも利用できることであり、障害が発生しても、安定したサービスが提供できるレベルを指す。可用性を高めるためには、コンピュータやネットワークのバックアップシステムの整備や縮退運転機能の導入、電源設備などのコンピュータ関連設備のバックアップを考慮する。

アは性能、イは可用性、ウは操作性、エはセキュリティを表す。求める答えはイとなる。

例題演習

クライアント管理ツールに備わっている機能のうち，業務に無関係なソフトウェアがインストールされていないことを確認するのに最も有効なものはどれか。

ア　インベントリ収集　　イ　遠隔操作
ウ　ソフトウェア配信　　エ　ライフサイクル管理

解答解説

ネットワークの運用管理に関する問題である。

インベントリ管理はパソコンなどのハードウエアやソフトウエアの構成情報である「インベントリ」を管理することである。インベントリにはハードウエアではＣＰＵの種類、メモリーやハードディスクの容量、ＬＡＮボードの種類などが含まれる。ソフトウエアでは、ＯＳや導入したアプリケーションの種類やバージョンなどがある。さらに、ネットワークのアドレスの

ように、ユーザーが設定した情報も含んでいる。

アのインベントリ収集によって業務に無関係なソフトウェアのインストールを確認することができる。求める答えはアとなる。

イの遠隔操作は、遠隔地から通信回線を経由して接続し、機器を操作することである。

ウのソフトウェアの配信は通信回線を経由してソフトウェアを接続機器に配布することである。

エのライフサイクル管理は計画・開発・運用・保守のシステムのライフサイクルを管理することである。

例題演習

パソコンの主記憶の効率的な使用に関する記述として，適切なものはどれか。

ア　各種のアプリケーションの処理中に異常終了が何回か発生したときは，デフラグメンテーションを実行する。

イ　主記憶領域の使用率を下げるために，デスクトップ上の利用頻度の少ないアイコンを削除したり，不要なウインドウを閉じたりする。

ウ　主記憶領域を確保するために，アーカイブ機能で書庫を整理する。

エ　データの記録と消去を頻繁に行った結果，処理速度が遅くなった場合は，スキャンディスクを実行する。

解答解説

パソコンの主記憶に関する問題である。

アのデフラグメンテーションは、ハードディスクなどに記録されたデータを適切に再配置して、データの構成を整理することであり、ディスクのアクセス速度の低下を改善する手段として利用する。アプリケーションの異常終了とは関係ない。

イのデスクトップ上のアイコンや開いているウィンドウは主記憶領域を使用しているため、削除したり、閉じたりすると使用率を下げることができる。求める答えはイとなる。

ウのアーカイブは、複数のファイルを１つにまとめてファイル容量を小さくし、ディスクの空き容量の確保に使用するものであって、主記憶領域の確保ではない。

エの処理速度が遅くなる原因は断片化であり、デフラグメンテーションで改善する。

例題演習

機密ファイルが格納されていて，正常に動作するＰＣの磁気ディスクを産業廃棄物処理業者に引き渡して廃棄する場合の情報漏えい対策のうち，適切なものはどれか。

ア　異なる圧縮方式で，機密ファイルを複数回圧縮する。

イ　専用の消去ツールで，磁気ディスクのマスタブートレコードを複数回消去する。

ウ　特定のビット列で，磁気ディスクの全領域を複数回上書きする。

エ　ランダムな文字列で，機密ファイルのファイル名を複数回変更する。

解答解説

機密ファイルの廃棄処理に関する問題である。

不要になったデータの廃棄に当たっては、重要データの漏洩を防止するため、厳重なチェックが不可欠である。セキュリティ上の必要性とデータ保全の必要性を考慮することが重要である。ＰＣの磁気ディスク上のデータの消去は、特定のビット列をディスクの全領域に上書き処理することによって読み出し不能にする。求める答えはウとなる。

アのデータの圧縮では、伸張の可能性が０にはならない。

イのマスタブートレコードを消去しても、静的に読み出すことが可能である。

エのファイル名を変更しても、ディスクから直接、データを読み出すことは可能である。

例題演習

磁気テープに保存されたデータの廃棄に関して，適切なものはどれか。

ア　管理上の保管期間が経過したデータがあったので，直ちに廃棄し，後で所定の手続をした。

イ　重要なデータの廃棄を外部業者に依頼したところ，以前に依頼したことのある業者だったので，廃棄方法などの確認はしなかった。

ウ　障害が発生して使用不能となったデータも，面倒だったが所定の手続に従って廃棄した。

エ　廃棄後は管理台帳から抹消して，機密保持のために当該データに関する記録が残らないようにした。

解答解説

データの廃棄管理に関する問題である。

アは廃棄処理後は直ちに廃棄記録簿を作成する。

イの外部業者を使用する場合は、業者との契約書を取り交わしてから委託する。

ウのデータの廃棄は所定の手続きを行って廃棄するという内容は正しい記述であり、求める答えはウとなる。

エの廃棄後も廃棄記録簿に管理上必要な情報を残す必要がある。

例題演習

遠隔保守サービスシステムに関して、適切な記述はどれか。

ア　遠隔保守方式によって、故障箇所の発見と切分けが短時間に行えるので、構成機器の冗長性は不要となる。

イ　地域差のない保守サービスを提供するには、保守対象システムの設置場所に、技術力の高い保守員の配置が必要となる。

ウ　通信回線を利用した遠隔診断機能によって、保守サービスセンタから保守対象システムの障害箇所を指摘することが可能となる。

エ　保守対象システムの障害記録情報を保守サービスセンタに伝送し、蓄積できるので、現地派遣による修理、保守が不要となる。

解答解説

遠隔保守サービスに関する問題である。

アの遠隔保守体制と構成機器の冗長性を同時に活用することによってシステムの信頼性と保守性を高めることが可能となる。遠隔保守システムは冗長性が不要というのは正しくない。

イの保守技術の高度化は、コンピュータシステムの保守支援システムを利用して実現するため、技術力の高い保守要員は必ずしも必要としない。

ウの遠隔診断機能による障害箇所の指摘は、遠隔保守サービスに関する適切な記述である。求める答えはウとなる。

エの現地派遣による修理保守は、保守内容によっては現地保守が必要なものも発生する可能性があり、不要とは言えない。

例題演習

予防保守に関する説明として，適切なものはどれか。

ア　サーバが何らかの原因でハングアップしたとき，自動的にシステムをリセットし再起動する。

イ　サーバにバッテリーを内蔵しておき，瞬断のような短時間の電圧低下が発生してもサーバがダウンしないようにしておく。

ウ　ハードディスク装置などの自動訂正済みエラーを分析することで，故障の前兆をとらえて部品をあらかじめ交換する。

エ　メモリモジュールをあらかじめ複数挿入しておき，どれかが故障した場合には，そのモジュールを論理的に切り離して起動する。

解答解説

予防保守に関する問題である。

アはシステム障害でシステムが自動的に再スタートする回復の手段である。

イは停電および電圧低下の防止に関する内容である。

ウの故障の前兆を予め把握し、部品を取り替える手段は予防保守の考え方である。求める答えはウとなる。

エは予備の手段を設け、システムの信頼性を高める考え方である。

例題演習

システムの保守に関する記述のうち，ＭＴＢＦを長くできるものはどれか。

ア　遠隔保守を実施する。

イ　故障発生箇所の臨時保守を実施する。

ウ　保守センタを１か所集中配置から分散配置に変える。

エ　予防保守を実施する。

解答解説

システム保守に関する問題である。

予防保守は、障害の発生を防ぐための保守であり、あらがじめ保守計画を立てて実施する。保守要員の確保も計画的に行えるため、効率の良い保守が実施できる。

アの遠隔保守は、点検や故障時の保守が遠隔からできる便利さが生じるが、必ずしも予防保守になるとは言えない。

イの臨時保守は故障発生時に行う保守であり、予防保守ではない。

ウの分散配置は、保守要員の技術的な能力問題等が発生する可能性があり、必ずしも予防保守ができるとは言えない。

エの予防保守は、運用時に予想されるシステム障害や運用問題に対して、トラブルを事前に予防する目的で技術支援や予防修正などを実施する。求める答えはエとなる。

例題演習

アプリケーションの保守に関する記述として，適切なものはどれか。

ア　テスト終了後は速やかに本稼働中のライブラリにプログラムを登録し，保守承認者に報告する。

イ　変更内容が簡単であると判断できるときは，本稼働用のライブラリを直接更新する。

ウ　保守作業が完了しないまま放置されるのを防ぐためにも，保守の完了を記録する。

エ　保守作業は，保守作業担当者によるテストが終了した時点で完了とする。

解答解説

アプリケーションの保守作業に関する問題である。

アのテスト終了後の作業内容は、登録、報告のみでなく、記録、文書化ができるまでの処理が必要である。

イの簡単な変更内容の処理も、記録、文書化までの処理を必ず行う。

ウの保守完了を記録する作業まで的確に行うことは適切な内容である。求める答えはウとなる。

エのテスト完了時点で保守作業完了の判断は誤りであり、必ず記録、文書化まで実施する。

例題演習

ソフトウェアの“修正保守”に関する説明として、適切なものはどれか。

ア　誤りの修正ではなく、より良いアルゴリズムの採用や出力メッセージの充実など、ソフトウェアの完成度を高めるために実施する。

イ　業務要件の変更や、ハードウェアやＯＳのアップグレードのような処理環境の変更に対応する。

ウ　本番稼働後の運用時に予想される問題に対して、トラブルを予防する目的で実施する。

エ　要求された機能が達成されていない場合、業務に支障が出ないように機能仕様書との不一致を修正する。

解答解説

ソフトウェアの修正保守に関する問題である。

システムが稼働後に、システムに潜んでいたバグが発見されたり、ユーザによるシステムの仕様の変更依頼があった場合、システムを修正しなければならないケースが生じる。この場合の作業を保守という。修正保守はプログラムの誤りや、処理・手続きなどの仕様の誤りであるバグによる不具合の修正作業である。優先順位が最も高く、早急に対処する必要がある。

アは改良保守、イは変更保守、ウは予防保守、エが修正保守であり、求める答えはエとなる。

例題演習

外部から供給される電源の瞬断時に，コンピュータシステムを停止させないために設置する装置はどれか。

ア　ＣＶＣＦ　　イ　ＵＰＳ
ウ　自家発電装置　　エ　予備電源受電

解答解説

ＵＰＳに関する問題である。

アのＣＶＣＦは、大型コンピュータの無停電電源装置で、自家発電機と組み合わせて使用する。商用電源と自家発電機の切替時のバックアップや不安定な発電機の電力の安定化の機能がある。

イのＵＰＳは、商用電源供給停止時に機能する無停電電源装置である。バッテリの直流をインバータにより交流に変換して供給する。求める答えはイとなる。

ウの自家発電装置は、自家用の発電機を使用して電力を供給する装置である。

エの予備電源受電は、予備用の電源を外部の商用電源で受電することである。

例題演習

ＴＣＯの説明として、適切なものはどれか。

ア　自社に導入した業務システムに対する開発コストとハードウェアのコスト
イ　ハードウェア及びソフトウェアの導入から運用管理までを含んだコスト
ウ　ハードウェア及びソフトウェアを整備・稼働させるまでのコスト
エ　ハードウェアやヘルプデスク、ユーザ教育などのテクニカルサポートに要したコスト

解答解説

ＴＣＯに関する問題である。

アは、開発コストとハードウェアコストであり、アップグレードや保守、教育研修などの費用が含まれていない。

イの内容はＴＣＯに関する記述である。求める答えはイとなる。

ウには、教育研修などの導入後にかかる様々な費用が含まれていない。

エには、ソフトウェアのアップグレードや保守の費用が含まれていない。

例題演習

コンピュータセンタの運用コストを実績課金法で部門別に配賦しょうとするとき、課金対象として適切なものはどれか。

ア　売上金額　　イ　磁気ディスク使用量
ウ　所属人数　　エ　生産高

解答解説

運用コストの課金方法に関する問題である。

コンピュータシステムの利用資源の割合に対して課金するのが当然であり、解答群の内容から考えて、磁気ディスクの使用量が適切である。売上金額、所要人数、生産高はコンピュータセンタの運用に必要なコストとは直接には関係ない。求める答えはイとなる。

例題演習

システム障害の早期発見のための対策として、最も適切なものはどれか。

ア　システムコンソールメッセージのロギング機能を設ける。
イ　主要な通信回線に予備ルートを設ける。
ウ　障害発生時にチェックポイントからの再開始機能を設ける。
エ　ファイルに障害が発生した場合のためにジャーナルファイルを設ける。

解答解説

システム障害の早期発見に関する問題である。

アのコンソールメッセージのロギング機能は早期発見の手段になる。求める答えはアとなる。

イの予備の通信回線は回線障害には有効である。

ウのチェックポイントは再開始機能の実現のために必要なものである。

エのジャーナルファイルは障害回復に必要な手段である。

例題演習

ホットスタンバイ方式に関する記述のうち，適切なものはどれか。

ア　待機系は，現用系が動作しているかどうかを監視していて，現用系のダウンを検出すると現用系が行っていた処理を直ちに引き継ぐ。
イ　待機系は，現用系に入力されるジョブを監視していて，処理量の大きいジョブが入力されると現用系に代わってこれを実行する。
ウ　待機系は，現用系の負荷状態を監視していて，現用系のオーバロード(過負荷状態)を検出するとオーバロードした分の処理を引き受けて実行する。
エ　待機系も現用系と同時に同じ処理を実行していて，現用系がダウンしても待機系が処理を完了する。

解答解説

ホットスタンバイ方式に関する問題である。

ホットスタンバイ方式は、コンピュータを２台以上用意し、その内１台がダウンしても他のコンピュータがその処理を即時に引き継げるように、プログラムやデータを生かした状態で待機させておく方式である。

アはホットスタンバイ方式、イは大負荷対策、ウは負荷バランス方式、エはデュアル方式である。求める答えはアとなる。

例題演習

システムが稼働不能となった際のバックアップサイトをウオームサイト，コールドサイト，ホットサイトの３種類に分類したとき，一般に障害発生から復旧までの時間が短い順に並べたものはどれか。

ア　ウォームサイト，コールドサイト，ホットサイト
イ　ウォームサイト，ホットサイト，コールドサイト
ウ　コールドサイト，ウォームサイト，ホットサイト
エ　ホットサイト，ウォームサイト，コールドサイト

解答解説

ホットサイトに関する問題である。

コールドサイトは、遠隔地の待機系の立ち上げに切り替えの時間が必要である。

ホットサイトは、故障発生時に遠隔地のシステムがプログラムやデータを素早く引き継ぎ直ちに起動できるように、バックアップファイルやデータの更新情報を転送しておく方式である。切り替えの時間は短い。

ウォームサイトはコールドサイトとホットサイトの中間の考え方である。

障害が発生してから復旧までの時間が短い順は、ホットサイト、ウォームサイト、コールドサイトの順になり、求める答はエとなる。

例題演習

遠隔地に待機系システムを設置し、バックアップファイルやデータの更新情報を転送しておく、災害発生時には、この待機系システムでファイルを復旧し、業務処理ができるようにする。このようなバックアップ形態はどれか。

ア　コールドサイト　　イ　ホットサイト
ウ　ミラーサイト　　エ　モービルサイト

解答解説

ホットサイトに関する問題である。

アのコールドサイトは、遠隔地の待機系の立ち上げに切り替えの時間が必要なものである。

イのホットサイトは、故障発生時に遠隔地のシステムがプログラムやデータを素早く引き継

ぎ直ちに起動できるように、バックアップファイルやデータの更新情報を転送しておく方式である。求める答えはイとなる。

ウのミラーサイトは、人気のあるウェブ・サイトや重要度の高いソフトウエアのアーカイブが置かれているＦＴＰサーバーにアクセスが集中して、サービスに支障をきたすことを避けるために設置する複製サイトである。アクセス頻度が高いデータや価値の高いデータの一部またはすべてのコピーを保持して二次配布サービスを行う。

エのモビールサイトは、携帯可能なコンピュータや公衆電話、携帯電話などを利用し、オフィス外からオフィス内のＬＡＮやコンピュータに接続して、コンピュータを利用することのできるサイトである。

例題演習

コンピュータシステムが災害や障害によって使用不能になった場合に，迅速に復旧させるための対策として，適切なものはどれか。

ア　遠隔地にバックアップ用のシステムを構築して，復旧に必要なプログラムやデータを定期的に送信しておく。
イ　許容範囲を超える電圧や周波数からシステムを保護するために，重要な機器をＵＰＳに接続する。
ウ　重要なデータを安全に保管するために，プログラムやデータファイルのコピーを禁止して，１か所に集めて厳重に管理する。
エ　ネットワーク機器の管理を設置場所に近い各部署に任せて，集中管理しないようにする。

解答解説

システムの復旧対策に関する問題である。

自然災害を含めた災害対策の課題は、システム資源を遠隔地に分散してバックアップ用のシステムを構築しておくことであり、定期的にデータの整合性の確保が必要になる。

アの遠隔地のバックアップ用のシステムにプログラムやデータを送信しておくのは災害対策の手段であり適切な記述である。求める答えはアとなる。

イのＵＰＳは電源の事故に対しては有効であるが、火災や地震、台風、洪水、落雷などの自然災害の対策には不十分である。

ウの重要データの一カ所での管理は、管理は容易であるが自然災害の対策には不十分である。

エの分散管理は、あるデータの管理はシステム内の特定の箇所にしか存在しないため、自然災害の対策として不十分である。

例題演習

障害発生時の原因追求に役立つデータとして、よく利用されるものはどれか。

ア　課金データ　　イ　コンソールログ
ウ　リカバリ用チェックポイントデータ　　エ　ファイルのバックアップデータ

解答解説

障害発生時の原因追及に関する問題である。

障害発生時には障害分析が必要になり、障害情報の収集が行われる。障害受付時点での情報として、障害発生時のログデータやダンプリストなどをもとに障害原因の分析が行われる。特に、システムコンソールに表示されるメッセージは多くの情報を提供する。

アの課金データはインターネットなどの利用料金に関する情報で、利用した日時、利用時間、サービス内容などをもとにアクセスするごとに計算され、日単位、月単位に集計する。

イのコンソールログは障害発生時の原因追及に役立つデータである。求める答えはイとなる。

ウのチェックポイントは検査又は再始動のために設けられたプログラム上の点で、プログラムの実行中に異常が発生しプログラムが中断したとき、この点を起点として再始動する。

エのバックアップはハードウェアやソフトウェアの故障や誤動作による破壊などの事故に備えて、データやプログラムの複製を作ることである。

例題演習

オンラインシステムの障害対策に関する記述のうち，適切なものはどれか。

ア　ジャーナルファイルやマスタファイルのバックアップファイルは，すぐに復旧処理ができるようにオリジナルファイルと同一の場所に保管する。

イ　トランザクションの処理が正常に終了できなかったときは，トランザクション開始直前の状態に戻すために，ロールフォワード処理を実行する。

ウ　マスタファイルと，一定時間ごとに作成したマスタファイル更新用のトランザクションファイルを用いて，システム障害発生直前の最新データを復元する。

エ　マスタファイルは，オンラインサービスの終了時にバックアップを取得するだけでなく，システムの特性に応じた時期にバックアップファイルを取得する。

解答解説

障害対策に関する問題である。

アのジャーナルファイルやマスタファイルのバックアップファイルをオリジナルファイルと同一の場所に保管するのは自然災害の地震や火災のことを考えると適切な方法ではない。

イのトランザクション開始直前の状態に戻すのはロールバック処理であり、ロールバック処理はシステム障害の復旧時には必要であるが、障害対策には不十分である。

ウのシステム障害の復旧には、チェックポイントでのマスタファイルとその時点でのログファイルが必要である。トランザクションファイルでは不十分である。

エのバックアップファイルは、オンラインサービスの終了時とチェックポイント時に必要であり、システムの特性に応じた時期に取得する。適切である。求める答えがエとなる。