# txt040112 開発モデルと構造化手法

## ① ウォータフォールモデル

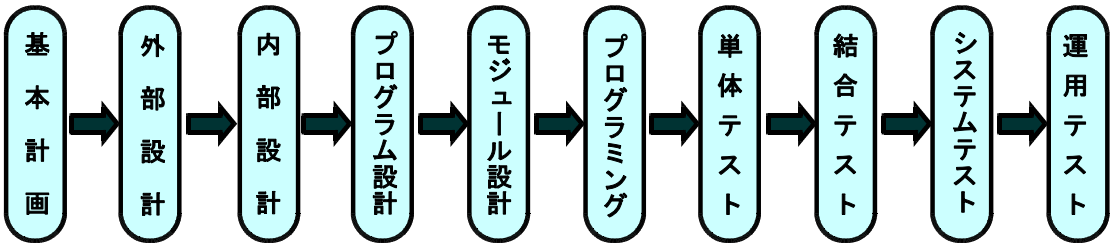
### ⓐ ウォータフォールモデルとは

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。

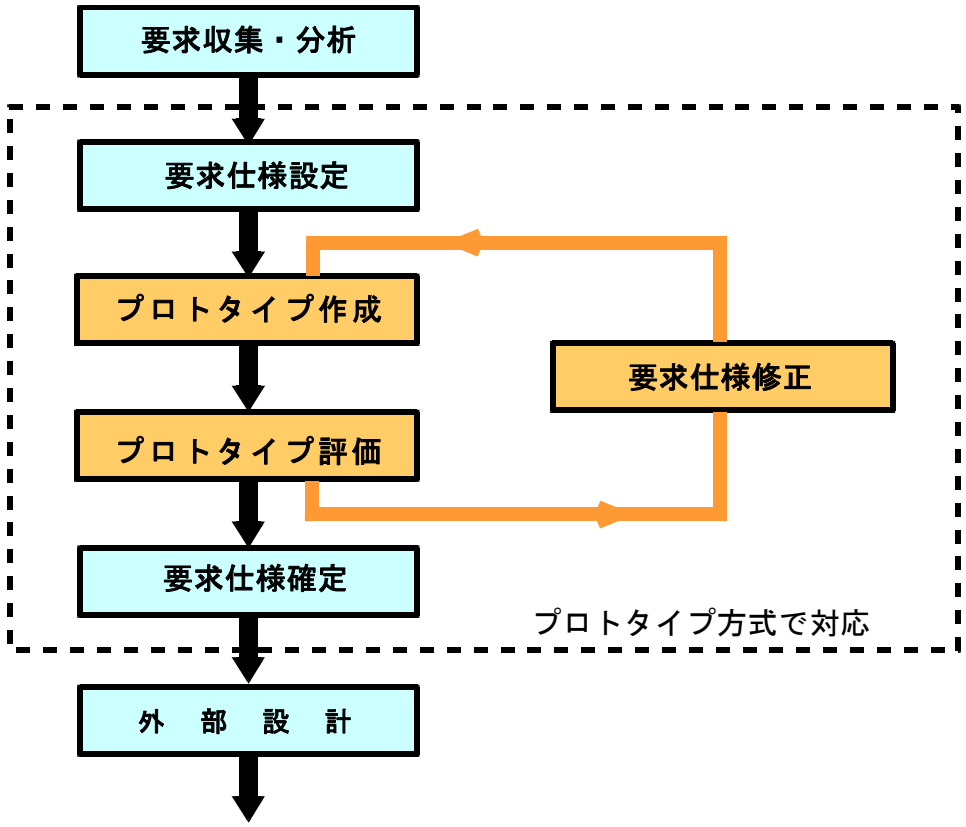
### ⓑ　スパイラルモデルの特徴

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

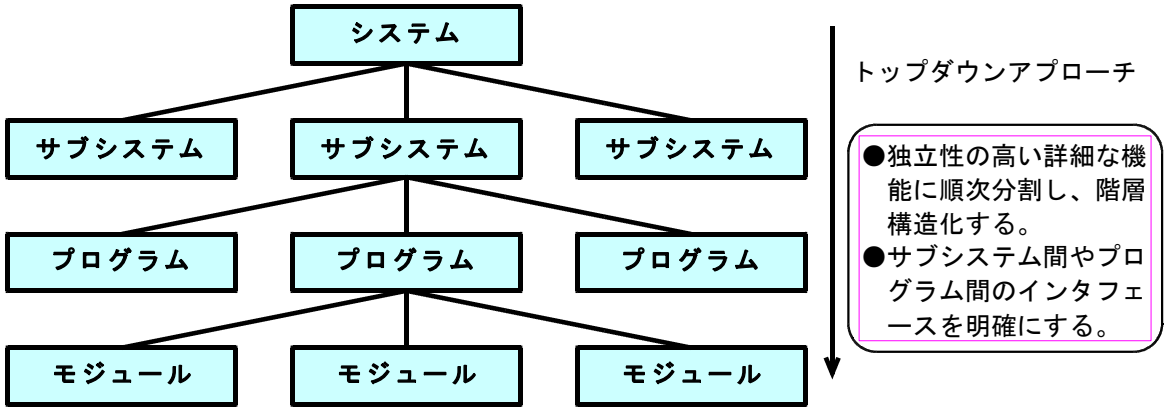
### ㋔　要員のスキル・経験

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

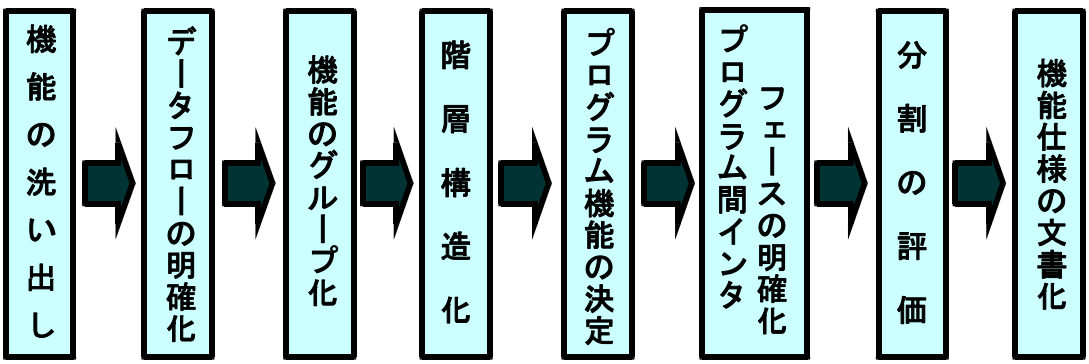
出力データに注目し、関係する入力データや処理内容を決定する。１つのプログラムの出力データがプログラム間のインタフェースとなる。

### 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。

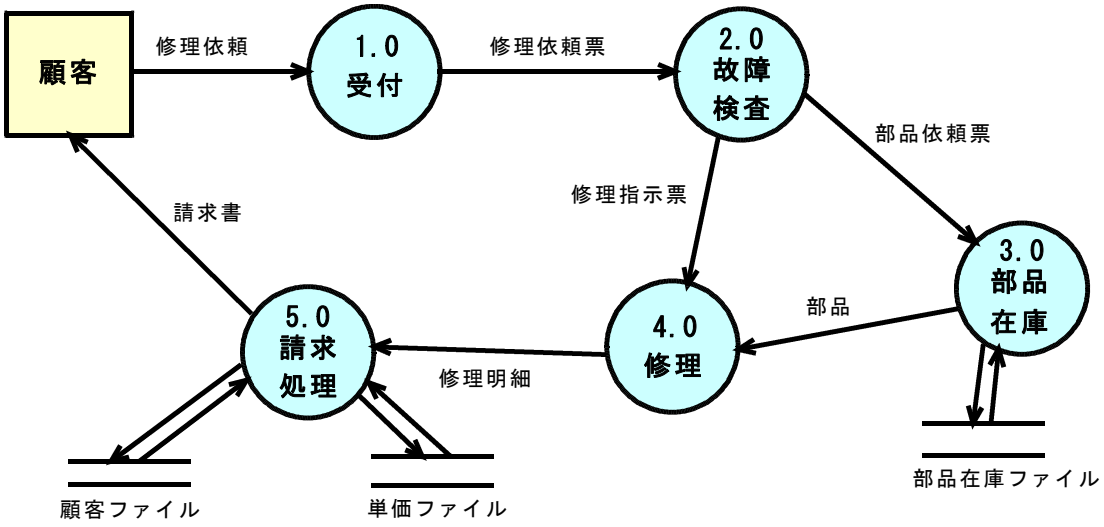
#### ㋒　流れ図の種類

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。

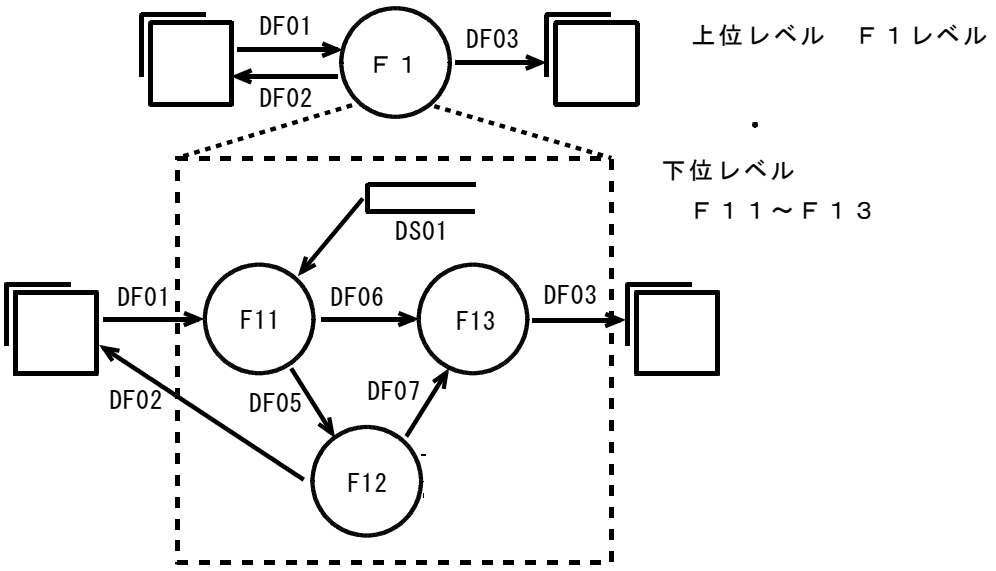
❸ **最下位レベルのＤＦＤに記載された各バブルごとにそれぞれのミニ仕様を作成する。**

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

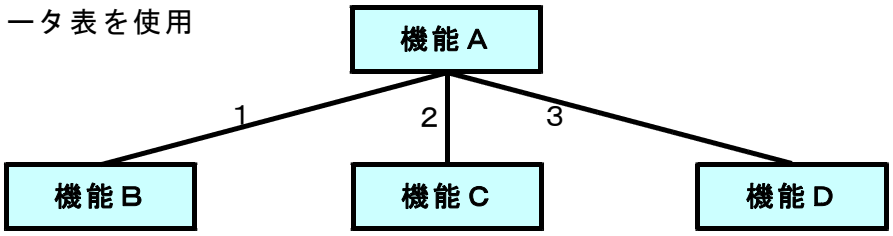
❶ パラメータの記述ができるため、機能（プログラム）間のインタフェースが理解しやすい。

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

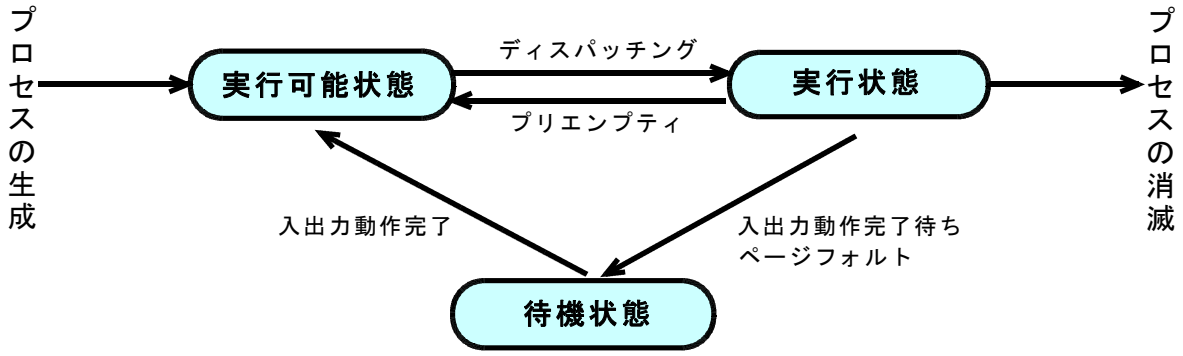
力ファイルを、処理欄に処理内容を、出力欄に出力ファイルを記述する。

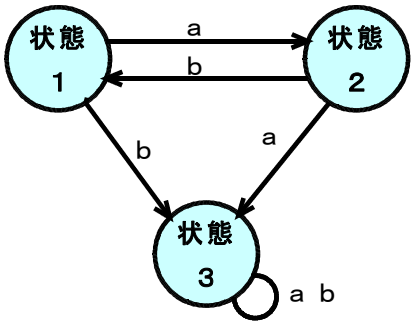
## ⓔ　状態遷移図・遷移表

### 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。

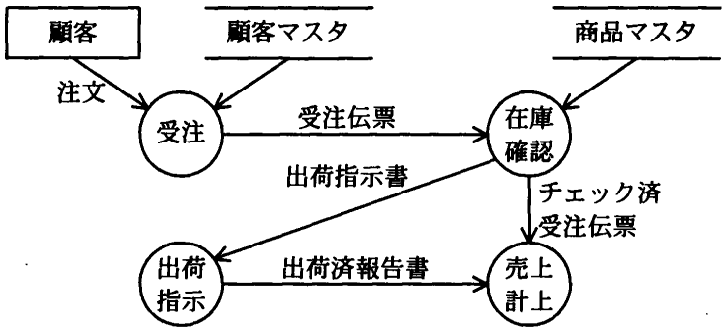
イのイベントは、状態変化を引き起こす出来事で、状態遷移図における遷移のきっかけになる出来事である。

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

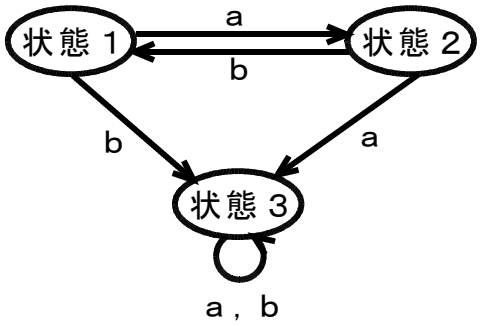
で表現する。タスクの状態遷移やリアルタイム処理の状態変化などを表現するのに利用する。

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。