ソフトウェア工学

Software Engineering

－　２００５年度版　－

福岡工業大学
情報工学部 情報工学科
柴田望洋

BohYoh Shibata
Fukuoka Institute of Technology

本資料について

◆　本資料は、2005 年度・福岡工業大学 情報工学部 情報工学科３年生の講義

『ソフトウェア工学』

の補助テキストとして、福岡工業大学 情報工学部 情報工学科 柴田望洋が編んだものである。

◆　参考文献・引用文献等は、資料の最後にまとめて示す。

◆　諸君が本資料をファイルに綴じやすいように、研究室の学生達が時間を割いて、わざわざ穴を開けるという作業を行っている（一度のパンチで開けることのできる枚数は限られており、**気の遠くなるような**時間がかかっている）。
必ずＢ５のバインダーを用意して、きちんと綴じていただきたい。

◆　本資料のプログラムを含むすべての内容は、著作権法上の保護を受けており、著作権者である柴田望洋の許諾を得ることなく、無断で複写・複製をすることは禁じられている。

◆　本講義では、以下に示すホームページ上の、各ドキュメントも参照するので、参考にされたい。

柴田望洋後援会オフィシャルホームページ　http://www.BohYoh.com/

■　本資料は、Microsoft 社のワープロソフトウェアである Microsoft Word 2003 を用いて作成した。

ソフトウェア工学とは

ソフトウェア工学（*software engineering*）という言葉が初めて使われたのは、1968 年の NATO 国際会議でのことでした（Naur, P. and B. Randell, Software Engineering, Report on a Conference Sponsored by the NATO Science Committee, 1969）。

この**ソフトウェア工学**は、小規模なソフトウェアは対象としません。すなわち、1 年生や２年生の『プログラミング言語』の講義で学習した、高々20〜40 行程度のプログラムは、基本的には対象外です。というのも、その程度のプログラムをどのように作成するかは、プログラミング言語レベルでの技術的な問題だからです。

大規模なソフトウェアの開発、複数の人間＝チームによるプロジェクトの推進、将来的に変更しやすいソフトウェアの開発、といったものが対象です。

ソフトウェア工学の定義は、まちまちですが、私独自の定義を示します。

※講義時に補足する内容をプリントに書き込みましょう。

ソフトウェア工学の定義（柴田望洋の定義）

品質の高いソフトウェアを効率よく開発・保守するための方法論および技法

なお、ここでの＜ソフトウェア＞は、プログラムだけでなく、仕様書やマニュアルなどのドキュメント（文書）なども含みます。

重要な点のみを考えても、

■ソフトウェア自身の品質の向上

■ソフトウェアの生産性の向上（どちらかと言えば開発・保守する人間側の問題）

の少なくとも二つを目指す学問であるといえます。

なお、ソフトウェアのライフサイクルの考え方、各種の分析法・設計法など、この分野は情報処理技術者試験でも頻繁に出題されます。

ソフトウェアの品質について

ソフトウェアの品質については、いろいろな研究者によって、さまざまな分類・定義が提唱されています。ここでは、品質の二面性について考えましょう。

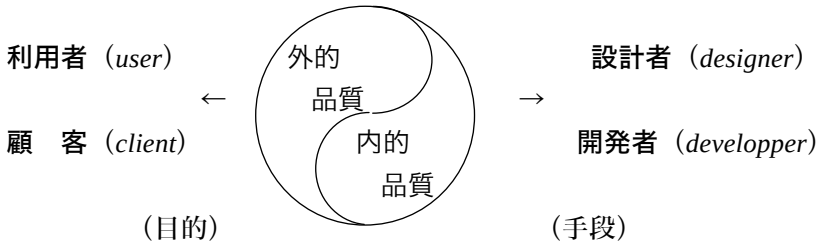
■ 外的品質 ■

スピード、使いやすさなどのように、ユーザがその有無を認識できるもの。

■ 内的品質 ■

モジュール性、プログラムの可読性などのように、どちらかといえば、開発者にしか分からないもの（利用者には直接的には関係ないもの）。

以上の二つの品質の関係を、以下の図に示します。



ソフトウェアは、「何よりも正確でなければならない」わけですから、最終的に要求されるのは外的品質です。もっとも、これは内的品質に支えられるものです。

■　ここで用いた**太極図**は道教で用いられる図です。陰と陽の面積が等しいこと、それらが裏表の関係にあること、境目（さかいめ）がないこと、不可分であること、などを表現するために曲線が用いられていることなどに留意しましょう。すなわち、円を半分に直線で割った図形（半円を二つ組み合わせた図形）とは根本的に違うものです。

■　自由課題：Illustrator などのソフトウェアを使って**太極図**を描いてみましょう。どのように描けばいいでしょうか？　試行錯誤してみると勉強になるかもしれません。

ソフトウェアの外的品質について

ここでは、外的品質の要因のうち、特に重要なものを解説します。

１．正確さ（*correctness*）

ソフトウェアが要求および仕様によって定義されたとおりに確実に仕事を行う能力

最も重要な要因であって、これを満足していないソフトウェアは論外です。さてさて、私の手元にある電卓で、

[1] [÷] [3] [=]

という計算を行うと、次のように表示されます。

０．３３３３３３

この計算結果は、数学的には明らかに**不正です**。正解は 0.333333‥‥すなわち数学的表記で表すと $0.\dot{3}$となるはずです。ちなみに、私の電卓で、

[×] [3] [=]

と計算を続けても、

０．９９９９９９

と表示されて 0.000001 の誤差が生じます。整数の除算・乗算すら正確に計算できない電卓を、なぜ皆さんは信用し利用しているのでしょうか。

また、逆に、もしも電卓が正確でないことが明らかであれば、そんなもの誰も使わないとも考えられるのですが…。

その答えは簡単です。私の電卓は“正確”です。たとえば「６桁の精度で計算する」とかいった仕様が決められており、その仕様の上において。

電卓は、仕様にのっとって、正確な計算を行うから、私も皆さんも、電卓を利用するのです。

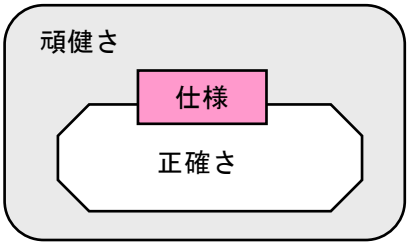
もちろん、まったく同じ計算を行っているのに、計算の度に違う答えを出力するとか、まちまちな精度で計算したりするのであれば、その電卓は正確とはいえません。

電卓の取扱説明書に、仕様が記載されていなければ、正確であるかどうかを判断することもできません。この電卓の例から、**仕様を決定することの重要性**も分かりますね。

２．頑健さ（*robustness*）

異常な状態においても機能するソフトウェアの能力

現実問題として、大規模なソフトウェアの仕様を完全に規定することは困難です。仕様に規定されていない状態においても、ソフトウェアが機能するかどうかは、正確さではなく**頑健性**（*robustness*）の問題です。



たとえば銀行のオンラインシステムにおいて、５箇所の別々のキャッシュコーナーで、存在しないはずの同一口座のキャッシュカード５枚を用いて、全く同時にお金をおろそうとしたらどうなるでしょう。

このようなケースにどう対処するか（たとえばキャッシュカード偽造と判断して、警報音とともにロックしてしまう etc ...）を、仕様として決めており、その通りに動作するのであれば、そのオンラインシステムは正確であるといえます。また、このようなケースにどう対処するかが仕様として決められていないときに、システムダウンするとか、無関係の口座の預金をおろすといった誤動作をせずに、それなりの対応をするのだったら、そのオンラインシステムは頑健であるといえます。

３．拡張性（*extendibility*）

ソフトウェアが仕様の変更に容易に適応できる能力

最近、ソフトウェアの『2000 年問題』が問題になりました。コンピュータに携わる人間は、“1 ビットでも節約したい”と考える人間が多いのです。多くのプログラマは、西暦年のデータに関して、下 2 桁だけを記憶させようと考えました。これだと 0〜99 の 100 種類の数が表現できれば十分であり、たったの 7 ビット（2^7＝128）で済むことになります。しかも、彼らは、自分の仕事に誇りを持っているものの、自信をもっていません。

「どうせ、我々の作ったソフトは、西暦 2000 年までもつはずないさ!!」…。

４．再利用性（*reusability*）

あるソフトウェアの全体または一部分が、どの程度新しいアプリケーション構築に再利用できるか

ニッサンのセドリックとグロリアはほとんど同じ。トヨタやニッサンに限らず、部品の大部分を流用して、複数の車種を作るのは当たり前となっています（マツダとフォードのような海外メーカとの提携もあります)。いろいろな部品を一から作らなくてすみますから。

日本語ワープロ「Word」の 80%を再利用して「一太郎」ができるのだったら、誰も苦労しないかな…。

５．互換性（*compatibility*）

ソフトウェア製品の組合せやすさ

ちょっとした日曜大工的な「工作」をすることを想像しましょう。作るものにもよりますが、カタログや規格表などから、いろいろな部品（パーツ）を選んで購入し、組み立てるだけで完成することもあります。

ソフトウェアで、そのようなことができるでしょうか。ソフトウェアを開発する際は、カタログショッピング的にパーツをチョイスして組み立てるだけ…といけばいいのですが、現実はそうではありません。

これまで全世界何十万人、何百万人のプログラマが開発してきたソフトウェアのパーツは、再利用性や互換性に乏しいことをあらわしているのかもしれません。

６．効率性（*efficiency*）

ハードウェア資源の使用法がどの程度効率的であるか

７．携帯性（*portability*）

さまざまなハードウェア環境やソフトウェア環境への移植のしやすさ

８．実証性（*verifiability*）

テストの準備やその手続きが行いやすいこと

９．統合性（*integrity*）

不当なアクセスや修正から構成部品を守るソフトウェアの能力

１０．使いやすさ（*ease of use*）

ソフトウェアの利用法習得、操作などが容易であること

■ 参考 ■ 浮動小数点数の演算について

計算の正確さについて、プログラムで確認してみましょう。**0.0001 + 0.0002 + 0.0003 + 0.9999 + 1.000** を求める二つのプログラムのどちらがより正確でしょうか？

■ **for** 文を整数で制御

```
/*
    0.0001, 0.0002, … 1.0000を表示・合計（for文を整数で制御）
*/

#include  <stdio.h>

int main(void)
{
    int    i;
    float  sum = 0.0;     /* 合計 */

    for (i = 0; i <= 10000; i++) {
        sum = sum + i / 10000.0;
        printf("%f¥n", i / 10000.0);
    }
    printf("合計値は%fです。¥n", sum);

    return (0);
}
```

■ **for** 文を浮動小数点数で制御

```
/*
    0.0001, 0.0002, … 1.0000を表示・合計（for文を浮動小数点で制御）
*/

#include  <stdio.h>

int main(void)
{
    float  x;
    float  sum = 0.0;     /* 合計 */

    for (x = 0.0; x <= 1.0; x += 0.0001) {
        sum = sum + x;
        printf("%f¥n", x);
    }
    printf("合計値は%fです。¥n", sum);

    return (0);
}
```

$10^{-20} + 10^{-19} + 10^{-18} + \cdots + 10^{18} + 10^{19} + 10^{20}$ を求める二つのプログラムです。

■ 小さい方から加える

```
/*
    10の-20乗 ＋ 10の-19乗 ＋ … ＋ 10の19乗 ＋ 10の20乗
*/

#include  <math.h>
#include  <stdio.h>

int main(void)
{
    int    i;
    double  sum = 0.0;           /* 合計 */

    for (i = -20; i <= 20; i++) {
        sum = sum + pow(10, i);
    }
    printf("合計値は%lfです。¥n", sum);

    return (0);
}
```

■ 大きい方から加える

```
/*
    10の20乗 ＋ 10の19乗 ＋ … ＋ 10の-19乗 ＋ 10の-20乗
*/

#include  <math.h>
#include  <stdio.h>

int main(void)
{
    int    i;
    double  sum = 0.0;           /* 合計 */

    for (i = 20; i >= -20; i--) {
        sum = sum + pow(10, i);
    }
    printf("合計値は%lfです。¥n", sum);

    return (0);
}
```

処理系によっても異なりますが、一般には、小さいほうから加えたほうが、より正確な値が得られます。どうしてでしょうか？

ソフトウェアの品質（その他の分類）

(ａ) INSTAC/STD による品質特性の分類

品質特性	副品質特性	
機能性 *functionality*	合目的性	*completeness*
	正確性	*correctness*
	セキュリティ	*security*
	互換性	*compatibility*
	接続性	*interoperability*
信頼性 *reliability*	無欠陥性	*non-defociency*
	誤り許容性	*error tolerance*
	可用性	*availability*
使用性 *usability*	理解性	*understandability*
	修得容易性	*ease of learning*
	操作性	*operability*
	対話性	*communicativeness*
効率性 *efficiency*	時間経済性	*time economy*
	資源経済性	*resource economy*
保守性 *maintainability*	修正容易性	*correctability*
	拡張性	*expandability*
	テスト容易性	*testability*
移植性 *portability*	ハードウェア独立性	*hardware independence*
	ソフトウェア独立性	*software independence*
	導入容易性	*installability*
	再利用性	*re-usability*

(ｂ) 片岡*による品質特性の分類

◎働き‥‥そのソフトウェアを用いるユーザに便益を与えるもの

合目的性・操作性・性能

◎正確さ‥‥そのソフトウェアが誤りなく働く度合い

信頼性・可用性・機密保護

◎適用範囲‥‥そのソフトウェアが利用できる範囲

時間的適用範囲

保守性・拡張性・互換性

空間的適用範囲

可搬性・再利用性・接続性

＊ 片岡雅徳『ソフトウェア・モデリング』，pp.42-61，日科技連出版社，1988

ISO / JIS による品質特性

ISO および JIS では、以下のように品質特性が規定されています。

品質特性	副特性
機能性（functionality）	合目的性（suitability）・正確性（accuarateness）・標準適合性（compliance）・接続性（interoperability）・セキュリティ（security）
信頼性（reliability）	成熟性（maturity）・障害許容性（fault tolerance）・回復性（recoverability）
使用性（usability）	理解性（understandability）・修得性（learnability）・運用性（operability）
効率性（efficiency）	時間効率性（time behavior）・資源効率性（resource behavior）
保守性（maintainability）	解析性（analysability）・変更性（changeability）・安定性（stability）・試験性（testability）
移植性（portability）	移植環境適応性（adaptability）・移植作業性（installability）・規格準拠性（conformnace）・置換性（replaceability）

■ 信頼性

ソフトウェアに要求された機能の実行が、所定の状況下で正常に維持されるかどうか。

回復性 … 故障しても容易に回復できる。

■ 効率性

処理に要する時間や資源を小さく抑えられるかどうか。

時間効率性 … スループット（単位時間当たりの仕事量）が大きい。

■ 使用性

操作や習得の容易さ。

運用性 … 運用管理が容易である。

■ 保守性

ユーザからの変更要求や障害への対応のしやすさ。ソフトウェア開発段階がいいかげであると、保守性も低くなります（読みにくく修正・変更しにくいソフトウェアとなってしまいます)。

解析性 … プログラムの解析が容易である。

■ 移植性

ソフトウェアを別のコンピュータ環境で使用するときに必要な変更の度合い。

第２種 平成 10 年度秋期 問 65

ソフトウェアの品質特性には、信頼性、使用性、保守性、移植性などがある。信頼性について説明しているものはどれか。

ア　操作や習得の容易さを表す。
イ　ソフトウェアに要求された機能の実行が、所定の状況下で正常に維持されるかどうかを表す。
ウ　ソフトウェアを別のコンピュータ環境で使用するときに必要な変更の度合いを表す。
エ　ユーザからの変更要求や障害への対応のしやすさを表す。

第１種 平成 12 年度 午前 問 69

ソフトウェアの品質特性のうち，保守性を表しているものはどれか。

ア　運用管理が容易である。　　イ　故障しても容易に回復できる。
ウ　スループットが高い。　　エ　プログラムの解析が容易である。

第１種 平成７年度 午前 問 69

ISO/JIS で規定されているソフトウェア品質特性に関する記述のうち、適切なものはどれか。

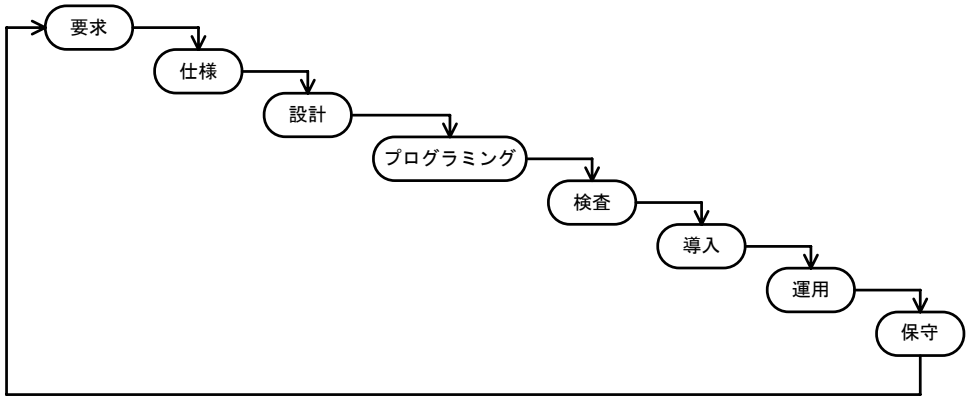
ア　使用性（usability）の副特性として、効率性（efficiency）が含まれる。
イ　使用性の副特性には、運用性（operability）は含まれない。
ウ　使用性の副特性には、試験性（testability）は含まれない。
エ　保守性（maintainability）の副特性として、移植性（portability）が含まれる。
オ　保守性は、ソフトウェアの開発段階とは関係しない。

ソフトウェアのライフサイクル

ソフトウェアの**ライフサイクル**（*software life-cycle*）とは、要求に応じてソフトウェアシステムが誕生し、稼働・運用されていく中で、保守が繰り返され、廃棄されるまでの期間を示します。

ウォータフォールモデル

ソフトウェアのライフライクルにおける各段階を、整然と水の流れるように推進するという考えでモデル化したのが下図に示す**ウォータフォールモデル**（*waterfall model*）です。すなわち、システム開発をいくつかの工程に分け、前段階の工程で得られた結果をもとに、順に開発・保守を進めていきます。



ウォータフォールモデル

(1) 基本計画

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。

(2) 外部設計

システムへの要求条件をもとに、システムとして必要な機能を定義する。

(3) 内部設計

システム構築上必要となる機能をプログラムに分割し、処理の流れを明確にする。

(4) プログラム設計

内部設計書に基づいて各プログラム内の構造設計を行う。

(5) プログラミング

詳細処理手順を設計、コーディングし、その修正を行う。

(6) テスト

テストを行う。

(7) 運用・保守

開発したシステムの運用を開始し、不都合やバグの発見、保守作業などによってシステムの修正を行う。

ウォータフォールモデルの欠点は以下のとおりです。

- 開発の初期段階に、曖昧さを排除しつつユーザの要求を取り入れることが困難である。
- システム開発を工程順に進めるので、後戻りすると開発効率が低下する。
- 後半の過程にならない限り、ユーザの要求を満たしているかどうかの判断やテストが行えない。

そもそもソフトウェアは進化し続けるものですから、この手法が必ずしもベストではありません。

第２種 平成６年度秋期 問 35

次の記述は、システム開発工程の作業内容を示したものである。開発手順に従って並べたものはどれか。

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　システム構築上必要となる機能をプログラムに分割し、処理の流れを明確にする。
c　詳細処理手順を設計、コーディングし、その修正を行う。
d　テストを行う。
e　内部設計書に基づいて各プログラム内の構造設計を行う。
f　システムへの要求条件をもとに、システムとして必要な機能を定義する。

ア　a-f-b-c-e-d　　イ　a-f-b-e-c-d　　ウ　a-f-c-b-e-d　　エ　a-f-e-b-c-d　　オ　a-f-e-c-b-d

第２種 平成 11 年度春期　問 56

次の記述は、システム開発工程の作業内容を示したものである。開発手順に従って作業内容を並べたものはどれか。

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　システム構築上必要となる機能をプログラムに分割し、処理の流れを明確にする。
c　詳細処理手順を設計、コーディングし、その修正を行う。
d　テストを行う。
e　内部設計書に基づいて各プログラム内の構造設計を行う。
f　システムへの要求仕様を基に、システムとして必要な機能を定義する。

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

第２種 平成 10 年度春期　問 60

大規模なアプリケーションを開発するとき、独立性の高い部分ごとに、設計、プログラミング、テストの開発工程を反復しながら完成度を高めて行く開発手法はどれか。

ア　E-R モデル　　イ　ウォータフォールモデル　　ウ　スパイラルモデル
エ　プロトタイプモデル

第２種 平成９年度秋期　問 60

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。

第２種 平成 12 年度春期　問 56

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るので、後戻りすればシステムの開発効率が著しく低下する。

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。

プロトタイプモデル

システム開発の早い段階で、目に見える形で利用者が要求を確認できるように試作品（プロトタイプ）を作成して、ユーザに試用してもらった評価を反映させながらプロトタイプを改良し、ユーザの要求や仕様を確定していく成長型の開発モデルが**プロトタイプモデル**です。利用者の要求を明確にして開発者にフィードバックし、利用者と開発者の認識の違いやあいまいさを取り除き、最終段階での食い違いを減らすことができます。利用者の参画意識の向上も期待できます。

特に、比較的小規模のアプリケーションの開発において効力を発揮します。

作成するプロトタイプは、あくまでも試験・評価用ですが、実際にコンピュータ上で動作するものでなければなりません。また、仕様が確定した段階で改めて本格的なシステムを開発することもあります（使捨て型）。

＜構築方法による分類＞
・使捨て型：目的終了後に試作品を破棄する。
・拡張型：試作品に肉付けして、本来のシステムに拡張する。

＜適用範囲による分類＞
・部分採用型：要求定義・外部設計段階で適用する。
・全面採用型：開発の全段階で適用する。

欠点は、以下の通りです。
・開発コストがウォータフォールモデル以上にふくらみがちである。
・スケジュールの調整が困難である。

第２種 平成７年度春期　問 63

システム開発の早い段階で、目に見える形で利用者が要求を確認できるように試作品を作成する開発手段はどれか。

ア　ウォータフォールモデル　　イ　オブジェクト指向　　ウ　クライアントサーバ
エ　スパイラルモデル　　オ　プロトタイピング

第２種 平成 10 年度秋期　問 60

プロトタイピングによるソフトウェアの開発に関する記述として、適切なものはどれか。

ア　ウォータフォールモデルとは異なる手法であり、両者を融合して使用することはできない。
イ　開発工程の早い時期にユーザの要求に合うかどうかを確認できるので、手戻りを少なくすることができる。
ウ　適用可能な範囲は、ユーザインタフェースの確認に限定される。

エ　プロトタイピングによって作成されたプログラムは、そのまま実稼動システムで使われることはない。

第２種 平成 6 年度秋期　問 36

プロトタイピングの特徴として適切なものを二つ選べ。

ア　実際に運用するシステムと同じものをプロトタイプでも実現しないと、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、ユーザに試用してもらい、評価と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。プロトタイプも同様に計算機上で動作できなくてよい。
エ　プロトタイピングでは、ユーザを開発過程に巻き込むことが難しいので、ユーザの参加意識の向上を図りたい場合、プロトタイピングの手法は適用すべきでない。
オ　プロトタイプはあくまで試作品であるので、仕様が確定した段階で改めて本格的なシステムを開発することがある。

第２種 平成 8 年度春期　問 59

ソフトウェア開発手法の一つであるプロトタイピングの特徴を記述しているものはどれか。

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行くため、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ため、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　ソフトウェアの開発における作業の分割とその管理を通常の契約関係とし、開発過程を管理する。
エ　ソフトウェアの性質を仕様が固定的で変更の必要のないものと、仕様の変更が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。
オ　大規模アプリケーションを独立性の高い部分に分解し、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。

第２種 平成 10 年度春期　問 61

プロトタイピングの特徴として適切なものはどれか。

ア　実際に運用するシステムと同じものをプロトタイプでも実現しないと、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、ユーザの参加意識の向上を図りたい場合、プロトタイピングの手法は適用すべきでない。

第２種 平成 11 年度秋期　問 56

ソフトウェ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行くため、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ため、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　ソフトウェアの性質を、仕様が固定的で変更の必要のないものと、仕様の変更が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解し、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。

第１種 平成９年度　問 61

プロトタイピングの説明として、正しいものはどれか。

ア　システムが要求仕様を満たしているかどうかを、基本設計段階で作成したチェックリストにより検証する。

イ　ソフトウェアライフサイクルの早い段階でシステムの一部を試作し、ユーザの要求に合うか否かを確認する。

ウ　プログラム開発の各工程で作成したものが、要求仕様に合っているかどうかを、開発メンバが集まって検証する。

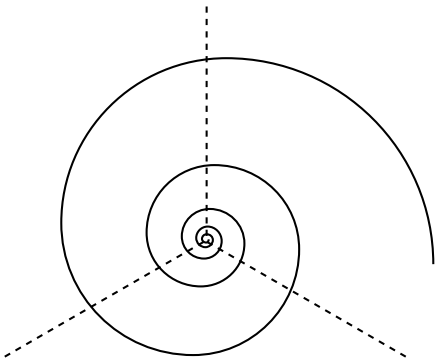
エ　プログラムの外部仕様を基に作成したテストデータを可能な限り組み合わせることによって、プログラムの機能を調べる。

スパイラルモデル

大規模なアプリケーションを開発するときなどに、独立性の高い部分ごとに、設計、プログラミング、テストの開発工程を反復しながら､反復・増殖的に完成度を高めて行く成長型の開発手法が**スパイラルモデル（螺旋モデル）**です。

各繰返しで、開発コストや品質などからリスクを評価し、リスクが小さくなるようなプロセスをとります。

- 開発の単位が比較的独立しているケースに適している。
- ウォータフォールモデルを部分的に踏襲している。
- 必要に応じてプロトタイプモデルを取り入れる。
- オブジェクト指向開発などでも用いられる。



第２種 平成 10 年度春期　問 60

大規模なアプリケーションを開発するとき、独立性の高い部分ごとに、設計、プログラミング、テストの開発工程を反復しながら完成度を高めて行く開発手法はどれか。

ア　E-R モデル　　イ　ウォータフォールモデル
ウ　スパイラルモデル　　エ　プロトタイプモデル

第１種　平成 10 年度　問 60

ソフトウェア開発手法の特徴に関する記述のうち、正しいものはどれか。

ア　ウォータフォールモデルは、要求分析、設計、プログラミング、テストの順に作業が流れていくので、多人数で行う大規模システム開発には不向きである。
イ　スパイラルモデルは、要求定義、設計、プロトタイピングを循環的に繰り返すので、類似したシステムを開発した経験がある場合に最適である。
ウ　成長モデルは、スパイラルモデルを改良した方法であり、機能分割と段階的な機能追加の繰返しによって、大規模システム開発に有効である。
エ　プロトタイピングは、実際に運用するシステムを作る前に試作品を作り、評価、改良を繰り返しながらシステムの仕様や性能を確定する方法である。

第１種　平成 11 年度　問 61

ソフトウェア開発プロセスモデルのうち、ウォータフォールモデル、スパイラルモデル、プロトタイピングとそれぞれの特徴の組合せとして、適切なものはどれか。

a　開発初期の段階での試作を通して、ユーザインタフェースの確定や、応答性などの性能確認を行い、後続段階での仕様変更による手戻りのリスクを防ぐ。
b　開発プロセスを繰り返しながら改良していく成長型モデル。各繰返しで、開発コストや品質などからリスクを評価し、リスクが最小となるプロセスをとる。
c　要求分析、システム設計、製造、テスト、運用・保守の順に逐次実行される一連の段階的工程からなる。

	ウォータフォールモデル	スパイラルモデル	プロトタイピング
ア	a	b	c
イ	b	c	a
ウ	c	a	b
エ	c	b	a

要求分析

ユーザ・顧客の要求を分析して、客観的なドキュメントにするのが、ソフトウェア開発の第一歩です。

キーボードから読み込んだ二つの実数値の和、積、差、商を求めて表示するプログラムを作成せよ。

上に示した問題の出題者の要求は簡単であり、どのようなプログラムを作成すればよいかが理解でき、何をすべきであるかが（ほぼ完全に）分かります。このような明瞭な要求内容は＜仕様＞です。

しかし、顧客の頭の中にある目的は、

「〜をしたい」

という漠然としたものである場合が少なくありません。

たとえば、レンタルビデオ屋の店長さんの頭の中に、

「不良会員（返却が滞りがちな会員）を除名したい。」

という目的があるとします。その目的は、

「不良会員リストを作成したい。」

という要求におきかえられます。ただし、この要求は＜仕様＞ではないことに注意しましょう。どのような入力情報を用いて、どういう出力情報が必要かということが、明確ではないからです。

要求は、処理に対する願望に過ぎません。

● 目 的 ●　不良会員（返却が滞りがちな会員）を除名したい

↓

● 要 求 ●　不良会員リストを作成したい

↓

● 仕 様 ● 不良会員とは、ビデオ返却延滞日数が多い会員のことである。
現在までの貸出・返却情報を用い、不良会員を５名までリストアップした一覧表を作成せよ。

要求分析における諸問題

要求分析では、次のような問題が起こりがちです。

(１) 顧客のニーズが不明確。
(２) 顧客のシステムに対する適用範囲の不理解性。
(３) 顧客のニーズを明確にドキュメント化する技法の欠如。
(４) 要求定義の不明確さによる潜在的バグの増大化。
(５) 要求定義の正確さを評価する基準の欠如。
(６) 検査の段階で検出された要求定義のミスの回復が不能。
(７) 顧客不在のままの要求定義が行われやすい。
(８) 要求定義のための系統的技法が確立されていない。
(９) 顧客と開発者の専門分野が異なり対応がとりにくい。

以下のことを忘れてはなりません。

★ 超重要 ★　要求は進化するものである。

■実例■　私が学生時代に“ レンタルビデオ” のシステムを開発したときのことです。

12 月下旬開店という店長から、開発の話しがあったのは 12 月上旬。わずか２週間で開発しなければなりませんでした。とりあえずは、以下の機能が欲しいとのこと。

・日常レンタル業務　（貸出し処理・返却処理・新規入会処理など）
・ビデオマスター保守（ビデオの登録・登録内容修正・削除・一覧表出力など）
・会員マスター保守　（会員の登録内容修正・削除・一覧表出力など）

当時私が開発に用いた処理系は Turbo Pascal。既に多くの流用可能なライブラリを開発済みでしたので、２週間という比較的短い開発期間も、技術的な点では、それほど苦ではありませんでした。

● 日常的レンタル業務のために、まずメニューを用意しました。ちょうど以下のような感じであり、（実際にはもっと複雑。ちなみに、数字キーによる選択だけでなく、カーソルキーによるメニュー間移動や、ファンクションキーによる選択なども可能です）。

１．貸出処理
２．返却処理
３．入会処理　　※入会処理終了後は自動的に貸出処理を行う
４．終了

作成途中の段階で、店長にみせたところ、このオープニングメニューは、いきなり<没>となりました。店長が言うには、以下のような処理が、この業界では常識ということでした。

■まずビデオのバーコードを読む　→　返却処理
■まず会員証のバーコードを読む
　□会員番号が登録済み　→　貸出処理
　□会員番号が未登録　→　入会処理

私は、笑いながら話を聞きながらも、心の中で「最初の打ち合わせの時に、ちゃんと言えよ」…。

●次は私の方が（？）悪い話。ソフトウェアの品質（頑健性）に関わる問題です。

年末にオープンしたレンタルビデオ店。私のシステムも順調に動作しています。ところが、新年早々、実家で寝ていた私に電話で嫌な知らせ。

「動作がおかしい。返却処理を行うと、すべて延滞日数が-364 日などと表示され、延滞料金はマイナスうん万円となる。」

そんなことはないはずだ、と私は思いました。自分の作ったソフトには自信がありますし…。

【後で分かった話】当時の NEC PC-9801 シリーズの内蔵カレンダー（日付や時刻を覚える機能）は、年をとらない仕様でした。たとえば 1985 年 12 月 31 日の翌日は、1985 年 1 月 1 日となっていたのです。年を繰り上げるためには、手動での時刻設定が必要だったのです。

私のソフトウェアは、「時間が逆戻りする」という４次元的なことは想定していませんでした。すなわち、**頑健さ**（p.5）に欠けていたのです。

●ソフトウェアを納入した後も、要求が次々と出てきました。一例を示します。

【その１】「ビデオの貸出ランキング表が欲しい」
この要求は一見簡単です。貸出処理を行う度に、各ビデオデータのカウンタを増やしていき、それをソート（順番に並べる）すればよいだけだ‥‥。と簡単にはいかないのですね。
ほとんどレンタルビデオを利用しない私は知らなかったのですが、人気のあるタイトルは、50〜100 本も入荷するとのこと。たとえば「ダイハード」が 50 本あるのであれば、その合計を貸出回数としなければならないのです。
私が最初に作成したシステムでは、それぞれ１本のビデオとして独立したデータとなっていたため、それらが同じタイトルであることを示すように<関連付け>をし、その合計回数を求める必要があり、その瞬間にソフトウェアの複雑さは、飛躍的にアップするのです。

【その２】「各会員の全貸出履歴が欲しい」
この要求は、各会員の「*年*月*日に、++++というタイトルのビデオを貸りた」という情報をデータベースに保存し、必要であれば、その情報を全て出力せよということです。
このような要求が、**いったんソフトウェアが完成した後に**出てきたところが困りものです。各会員のデータとしては、入会日、氏名、住所、電話番号、生年月日、入会時の身分証明書の種類などを格納しています（はっきりと覚えていませんが、2048bytes 位で設計していました）。そこには、履歴を格納するような余裕はありません。さらに困ったことに、２〜３回くらいだけ借りて、それっきりの会員もいれば、年に百本以上も借りるというオタッキー（？）な会員もいるとのこと。すなわち、各会員のデータ容量を固定できないのです（たとえば、２百本分の履歴領域を設定すれば、それでは足りない会員も出てくる一方、大部分の会員のデータは、すっからかんということになってしまいます！！）。

結局は、ページングの手法を用い、当初のデータ形式とほぼ完全な互換性を持たせつつ解決することになりました。

…以上は、ほんの一例です。私は学生時代に、数多くのソフト開発をしました（実際に覚えきれないくらいのソフトウェアを開発しました）。レンタルビデオのシステムよりも技術的に大変なものもありましたが、レンタルビデオシステムは、十分に要求を顧客から引き出さないうちから、設計・開発に着手したことによる弊害が多大であったという点で、最も印象に残っています。

分析から設計へ

ここで、仕様の指示の仕方について考えます。

宣言的（*declarative*）

「10 個の整数 n_1、 n_2、 ･･･、 n_{10}を昇順に並べかえよ。」

※並べかえるアルゴリズム（手順）は、数多くのものが考案されている（単純挿入ソート、単純選択ソート、単純交換ソート、クイックソートなど)。どれを使えば良いか分からない。

手続き的（*procedural*）

「10 個の整数 n_1、 n_2、 ･･･、 n_{10}の合計を求めよ。」

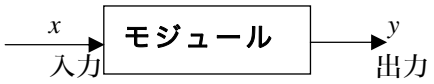
※10 個の値の合計を求める方法は、ほぼ一意的に決定できる。

したがって、次のように言うことができます。

設計作業とは、入力情報から出力情報への変換を**手続き的に**表現することである。

数学的表現

$y = f(x)$



p.20 の要求について

● 目 的 ●　不良会員（返却が滞りがちな会員）を除名したい

↓

● 要 求 ●　不良会員リストを作成したい

↓

● 仕 様 ● 不良会員とは、ビデオ返却延滞日数が多い会員のことである。
現在までの貸出・返却情報を用い、不良会員を５名までリストアップした一覧表を作成せよ。

設計の段階に入ると、“延滞日数が多い”という定義が曖昧であることが分かります。1 本のビデオを５日延滞した会員と、５本のビデオを１日ずつ延滞した会員とで、どちらの延滞日数が多いと言えるでしょうか。

････「延滞日数が多い会員を選べ」という仕様が曖昧であることが、設計の段階で明らかになりました。分析・要求定義を行った技術者に問題を差し戻すことになります。

外部設計と内部設計

システム設計（*system design*）は、分析によって得られた結果を、コンピュータ上で有効に利用できるようなソフトウェアを実現できるようにするための作業です。主に、外部設計と内部設計に分けられます。

まず、これらについて大まかに理解しましょう。

外部設計（outline design）

名前の通り、概要を設計します。

ユーザ要件の収集分析結果をもとに、システムの機能を確定します。また、その機能を実現するために必要な基盤（システム構成など）を明確にします。この際、コンピュータシステムの内部に関わりなく、ユーザの要求･意見を反映しなければなりません。プログラムの構造を決定し、その詳細機能を洗い出します。

具体的には、サブシステムの定義と展開、画面設計、報告書設計、帳票設計、コード設計、論理データ設計、外部設計レビューを行います。

そのため、具体的には、データ項目の洗い出し、データの関連性の分析、ファイル仕様の作成などを行います。

内部設計（detail design）

名前の通り、詳細を設計します。

外部設計がユーザ側から見た設計であるのに対し、外部設計書をもとに、どのようにコンピュータに処理させるかを設計する、開発者側から見た設計です。

システム構成上必要となるすべての機能をプログラムに分割し、プログラム間の処理の流れを明確にします。

具体的には、機能分割・構造化・物理データ設計、入出力設計、内部設計レビューを行います。そのため、具体的には、ファイルのアクセス時間やその容量を見積もったり、レコードレイアウトの決定を行ったりします。

顧客データベースの項目に、「会員番号」「氏名」「電話番号」が必要だというところまでが外部設計で、それぞれの項目を何バイトにするか、どのようなファイル形式として実現するかは内部設計で行います。

※　**レビュー**とは、複数の関係者によって、設計書やソースプログラムなどの曖昧な点や問題点などを洗い出すために行われる討議や評論のこと。

第２種 平成 6 年度秋期　問 37

システム開発工程のうちの外部設計における作業内容として、適切でないものはどれか。

ア　会話処理を実現するための画面遷移や画面レイアウトを設計する。
イ　コードを付与する対象を選定し、コード付与対象ごとにコード表を作成する。
ウ　プログラムの構造を決定し、その詳細機能を洗い出す。
エ　報告書の出力媒体の選択や報告書レイアウトの設計を行う。
オ　ユーザのシステム化要求を明確化するため、業務フローを整理、確認する。

第２種 平成 10 年度春期　問 59

システム開発工程の外部設計で行う作業として、適切なものはどれか。

ア　業務分析　　イ　帳票設計
ウ　テストケース設計　　エ　物理データ設計

第２種 平成 11 年度春期　問 57

システム開発工程の外部設計で行う作業として、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

第２種 平成 8 年度春期　問 61

システム開発の手順における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 要求定義　　オ 論理データ設計

第２種 平成 8 年度秋期　問 52

内部設計段階の物理データ設計において、実施する項目を二つ選べ。

ア　アクセス時間と容量の見積り　　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
オ　レコードレイアウトの決定

外部設計

コード設計

コードとは、本来の名称を識別するための記号、略号、符号、暗号のことである。その機能としては、

□識別機能□

情報を一意で表現し、他のものと区別できる。

□分類機能□

たとえば地域別、部門別などといった感じで、まとまりごとに分類できる。

□配列機能□

順番に並べることができる。

□チェック機能□

入力の誤りなどを防ぐことができる。

があります。

コード設計の留意事項として、

□利用範囲と使用期間を決定しておく　□扱いやすさと標準化

□すべての業務での共通性の確保　□体系化

□拡張性　□明瞭性

などが必要です。

なお、コード化は数字を主体にすべきであり、漢字は複雑なので使用すべきではありません。

ちなみに《JIS X0303 性別コード》では、

1 男

2 女

と定義されています。

また、《JIS X0401 都道府県コード》では、

01　北海道

02　青森県

(中略)

40　福岡県

(以降省略)

と定義されています。

この他にも、大学コードなどが JIS で定義されています。

なお、いったん定義したコードを変更することは好ましくありません。ただし、項目・分類の増減の可能性について十分な考慮が必要です。

コード化の種類

代表的なコード化の種類を紹介します。

(1)順番コード（sequential code）

データを 50 音順、入力順などの一定の順序に並べ、一連の番号を付ける方法です。

01　北海道
02　青森県
03　岩手県

(2)区分コード（block code）

データを、グループに分類し、各グループに対して順番コードを割り振る。桁数は短くて済むが、データの追加を考慮して、空き番号を確保しておく必要がある。

01〜39　　総務課
40〜69　　教務課
70〜89　　学生課

(3)桁別コード（group classification code）

コードの各桁に意味をもたせ、大分類、中分類、小分類といった感じで与えるコード。

1000　　本社　　大分類
1100　　総務部　　中分類
1110　　人事課　　小分類
1111　　採用係　　項目

(4)表意コード（mnemonic code）

17 インチテレビを TV17 とするように、データの内容を直接あるいは間接的に意味する文字や数字を使ったコード

(5)10 進コード（decimal code）

0、 1、 …、9 の 10 進数字だけを使ったコードで、図書館の図書の分類などに使われているコードです。

100　哲学
200　宗教
300　社会科学
320　　法律

第２種 平成 8 年度秋期　問 62

コード設計とそのコードの維持管理について、適切な記述はどれか。

ア　コード化は数字を主体にすべきであり、漢字は複雑なので使用すべきではない。
イ　コードが変化することは必然であり、コードブックの整備、維持が重要である。
ウ　コード自体によって、その意味が理解できることが望ましいので、けた数は長くするほどよいコードといえる。
エ　コードはデータの分類の容易さを目的として付与すべきであるが、追加や拡張性については考慮すべきではない。
オ　コード付与規則によって、コードの一元性を保持することは十分に可能なので、更にコンピュータによる管理機構は不要である。

第２種 平成 11 年度春期　問 60

コード設計とそのコードの維持管理について、適切な記述はどれか。
ア　コードが変化することは必然であり、コードブックの整備、維持が重要である。
イ　コード自体によって、その意味が理解できることが望ましいので、けた数が大きいほどよいコードといえる。
ウ　コードは数字を主体にすべきであり、漢字は複雑なので使用すべきではない。
エ　コードはデータの分類の容易さを目的として付与すべきであるが、追加や拡張性については考慮すべきではない。

第２種 平成 10 年度春期　問 67

市町村に対して、区は 101～199、市には 201～299、町村には 301～799 のコードが付けられている。このようなコード付けの分類を何というか。

ア　10 進コード　　イ　区分コード　　ウ　けた別コード　　エ　順番コード

チェックディジット：コードのチェック

入力を誤って、不正なコードを入力してしまうと、システムの動作に影響を与えかねません。コードの誤りをチェックする方法の一つがチェックディジットを用いた方法です。

チェックディジットとは、本来のコードに対して余分に付加するチェック用の数字です。その数字の与え方として、よく使われるのがモジュラス 11 と呼ばれる方法です。

モジュラス 11

モジュラス 11 と呼ばれる算法では、次のようにしてチェックディジットを求めます。

(a)　コードの各桁に重みを掛け、その合計を求めます。

(b)　合計値を 11 で割って剰余を求めます。

(c)　11 から剰余を引いて得られた値をチェックディジットとします。

第２種 平成 12 年度春期　問 60

コード化におけるチェックディジットの利用に関する記述として、適切なものはどれか。

ア　コード誤りの訂正にも有効である。
イ　コンピュータへの入力から出力までの処理内容の誤りを発見するのに有効である。
ウ　数字以外の文字を含むコードにも適用することができる。
エ　すべてのコード誤りを検出することができる。

第２種 平成 11 年度秋期　問 60

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　チェックディジット
ウ　チェックポイント　　エ　デシマルコード

基本 平成 13 年度春期　問 56

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。

第２種 平成 8 年度春期　問 66

4 けたのコード $N_1N_2N_3C$ 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けたのコードの□にあてはまる数字はどれか。

81□6

ア　0　　イ　2　　ウ　4　　エ　6　　オ　8

第２種　平成６年度秋期　問 38

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、この数値で検査文字を定め、データの最終けたに付加することによって、入力データの検査を行う方法がある。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。ここで、データの各けたに割り当てる係数は、先頭から 4、3、2、1 とする。

［規則］

(1) 各けたの数値と割り当てた係数との積の和を求める。

(2) (1) で求めた値を 11 で割った余りを求める。

(3) (2) で求めた余りの数字を検査文字とする。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　　オ 9

第２種 平成 12 年度秋期　問 55

与えられたデータから一定の規則に従って、数値を算出する。この数値を基に検査文字を定め、データの最終けたに付加することによって、入力データの検査を行う方法がある。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積の和を求める。

(3) (2)で求めた値を 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。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

第１種 平成 10 年度　問 64

次の方式によって求められるチェックディジットの付加結果はどれか。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ごとの積を求め、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後に付加する。

ア 73940　イ 73941　ウ 74944　エ 73947

ヒューマンインタフェース設計

画面設計では、ヒューマンインタフェースを考慮して行う必要があります。入出力項目の配置、入力順序、画面遷移など利用者の立場に立って設計します。操作方法はできるだけ標準化されていることも重要です。

□適切にレイアウトされている。

できるだけ左から右へ、上から下へと視線が移動するようにレイアウトする。

重要な項目や、入力項目は、区別できるように色や罫線を与える。また、関連する項目はグループ化する。

メニューの項目が多い場合、グループ化したり、階層化したりする。

文字だけでなく、アイコンなども利用する。

□適切なガイドやメッセージなどが表示される。

利用者が迷わないように、ガイドやメッセージを表示したり、ヘルプ機能をもたせる。

エラーメッセージを表示する際は、ユーザにエラーの原因だけでなく、その後の対処方法を示すとよい。

□操作性が統一されている。

画面によって表示や操作が異なると使いにくくなる。

利用者の操作回数を減らすために、既定値を与えるなどの工夫をする。

□複数の操作法を提供する。

たとえば、マウスによる初心者用の操作だけでなく、キーボードのショートカットによる上級者用の操作を提供する。

第２種 平成７年度春期　問 66

外部設計や内部設計の工程において､画面の設計を行う場合の留意事項として適切でないものはどれか。

ア　画面遷移ではメニューによる段階的選択だけでなく、習熟者向けの直接選択を考慮する。
イ　画面上の入力項目は、“　　”や［“　　”］で囲うなど、入力するフィールドであることを強調する。
ウ　参照する項目が、左から右、上から下に配置されるように画面レイアウトを設計する。
エ　処理を完結させるため、データ入力の中断や前画面への戻りなどはできないように設計する。
オ　タイトルやメッセージの表示位置を統一するなど、画面レイアウトの標準化を図る。

第２種 平成７年度秋期　問 66

ヒューマンインタフェースの設計時に留意すべき事項として、好ましくないものはどれか。

ア　エラーメッセージでは、使用者にエラーの原因とその後の対処方法を示す。
イ　システムの中にある情報や計算によって求められる情報であっても、必ず確認のため再度入力させる。
ウ　処理時間が長くなる場合は、処理の進行状況を使用者に知らせるためのメッセージを定期的に表示する。
エ　操作を覚えやすくするために、使用する用語、フォーマット及び操作手順に一貫性をもたせる。
オ　ヘルプ機能を用意し、操作方法やエラー時の復旧方法を提示する。

第２種 平成９年度秋期　問 65

メニューの設計に関して、適切な記述はどれか。

ア　ヘルプ機能があれば、目的のメニューを表示させるためのショートカット（近道）手段は不要である。
イ　ポップアップメニューやプルダウンメニューの構造は、項目数が多くなった場合でも単層とするのが望ましい。
ウ　メニューの構造を考えるとき、作業の種類を考慮する必要はない。
エ　利用者がコマンドを覚えなくても済むようなメニュー項目を設ける。

第２種 平成６年度秋期　問 63

対話型処理システムのヒューマンインタフェースを向上させるため、ファイル、プリンタなどの資源やコマンドなどを分かりやすい図形やパターンで表現し、画面に表示する。この図形やパターンを何というか。

ア　アイコン　　イ　カーソル　　ウ　スクロール
エ　マウス　　オ　メインメニュー

第２種 平成 10 年度秋期　問 67

GUI 部品の一つであるラジオボタンの用途として、適切なものはどれか。

ア　幾つかの項目について、それぞれの項目を選択するかどうかを指定する。
イ　画面をスクロールする。
ウ　コマンドや機能を起動する。
エ　互いに排他的である幾つかのオプションから一つを選ぶ。

基本 平成 13 年度秋期　問 50

GUI の設計で、排他的な複数の項目から一つだけを選択させたいときに使うコンポーネントはどれか。

ア　スクロールバー　　イ　スライダ
ウ　チェックボックス　　エ　ラジオボタン

第２種 平成 12 年度春期　問 68

システムの外部設計が完了したとき、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　処理方式　　エ　フローチャート

第２種 平成 11 年度春期　問 61

アプリケーションプログラムのエラーメッセージを設計する際の留意事項として、適切なものはどれか。

ア　エラーの対策内容は省略し、エラー内容、事実だけを表示すべきである。
イ　システム開発者がエラーを究明するために必要な情報だけを表示すべきである。
ウ　短いほど良いので、略号やエラーコードをそのまま使用して表示すべきである。
エ　利用者が何をすべきかを簡潔かつ詳細かつ正確に、肯定的に表示すべきである。

分析技法

要求仕様の分析法として、HIPO、状態遷移図、DFD などがあります。

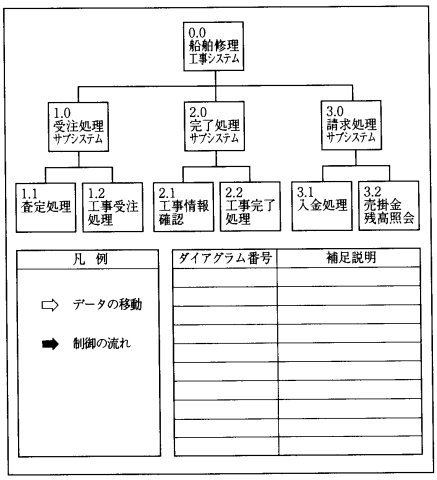
HIPO：Hierarchy plus Input-Process-Output diagram

構成モジュールの**階層構造**（Hierarchy）と、各モジュールを**入力・処理・出力**（Input-Process-Output）の形で表現したもの。入出力と処理ステップの関係を明確に表現できます。

□ 階層構造

図式目次（VTOC：Visual Table Process Output）で全体の機能構成を示し、プログラムやモジュール間の上下関係を階層的に表現します。

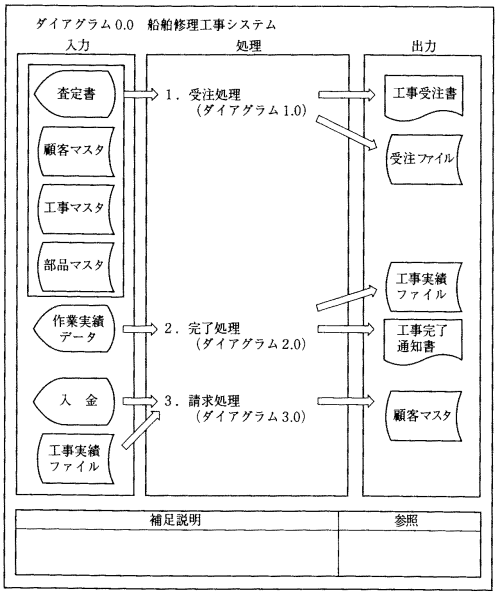
右図が、その一例です。



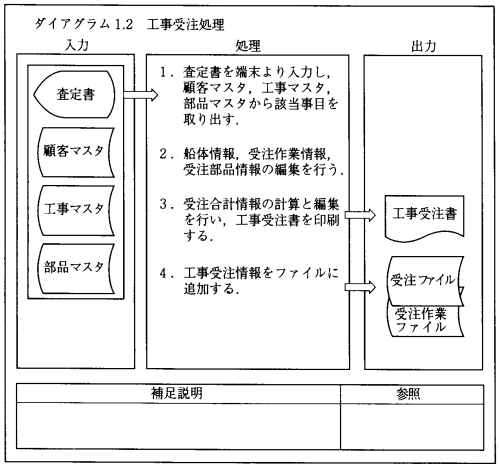
図式目次

□ 入力・処理・出力

総括ダイアグラム、詳細ダイアグラムによって、プログラムやモジュールへの入力・処理・出力を図で表現します。



総括ダイアグラム



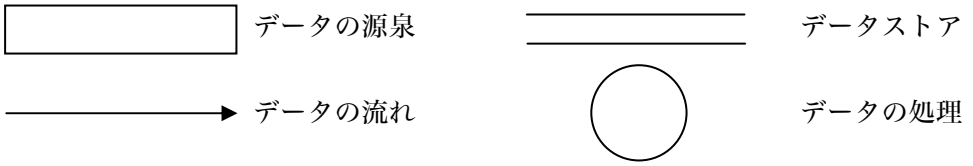
詳細ダイアグラム

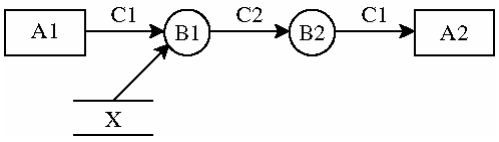
状態遷移図

時間経過や状態変化に着目して、システムの動作を記述するときに便利な表現方法。楕円で実体を表現し、矢印で状態の遷移を表現する。

データフロー図／DFD：Data Flow Diagram

データの流れを中心にシステムを表現する方法。情報の流れ（データフロー）を矢印、処理をボックス、データストア（データが蓄積されている状態）を２本の太線、外部（データの発生源または行先）を四角で表す。





バブルチャート

DFD と同じ。ただし、処理を円でなくボックスで表す。

流れ図：Flow Chart

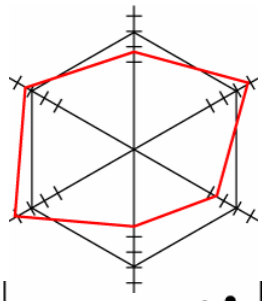
処理の順序やデータの流れを図に表現したもの。

ＮＳチャート

ナッシュとシュナイダーマンによって考案された表記法です。論理の流れを、流れ線を用いずに、プログラムの全体構造を階層構造の形で表現します。構造化ダイアグラムとも呼ばれます。

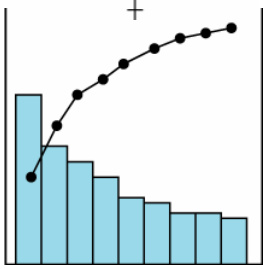
（参考：以下の二つは構造化分析とは関係ありません）

レーダーチャート



円に対して中心点から放射状に線を引く。1 本の線で一つの項目を表し、円との接点を基準とし、基準点との割合で線上に点を打ち、点と点を線で結んだもの。財務分析などでよく使われる。

パレート図



多数の管理項目の重要度を判定するため、大きさの順に項目を並べ、その累積和をグラフに記入したもの。

第２種 平成 8 年度秋期　問 61

構造化設計技法の一つである HIPO に関する記述として、適切なものはどれか。

ア　図式目次とデータフロー図で表現される。
イ　図式目次には、処理ブロック間で受け渡す制御情報を矢印とともに記述する。
ウ　図式目次は、プログラムの全体的な機能を示し、処理ブロックに記述される番号は処理の順序を示す。
エ　選択や繰り返しを表現するために流れ図の記号を用いる。
オ　入出力と処理ステップとの関係を明確に表現できる。

第２種 平成 13 年度秋期　問 47

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

第２種 平成 8 年度秋期　問 60

構造化分析手法の一つで、データフロー、処理（機能）、データストア、外部（データの発生源／行先）の記号を用いて、機能とデータの流れを表現するものはどれか。

ア　DFD　　イ　E-R 図　　ウ　NS チャート
エ　状態遷移図　　オ　ワーニエ図

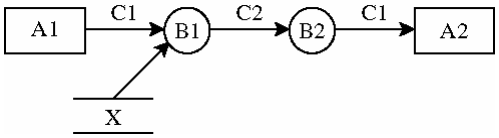
第２種 平成 10 年度秋期　問 61

プロセスモデルの表現に DFD が使用される。DFD の要素でないものはどれか。

ア　エンティティ　　イ　外部（データの発生源／行先）
ウ　処理　　エ　データストア

基本 平成 13 年度秋期　問 46

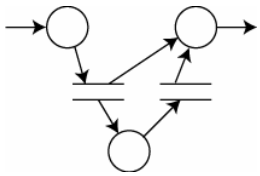
次の DFD で、記号 X が表すものはどれか。



ア　データ源泉　　イ　データストア　　ウ　データフロー　　エ　プロセス

第２種 平成 9 年度春期　問 59

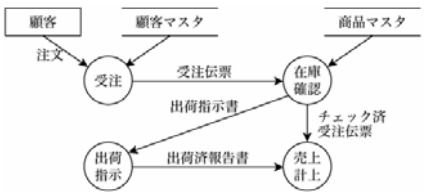
次の図は構造化分析法で用いられるデータフロー図の例である。図中の＿＿＿が表すものは何か。



ア　状態　　イ　データストア　　ウ　データフロー　　エ　プロセス

基本 平成 13 年度春期　問 44

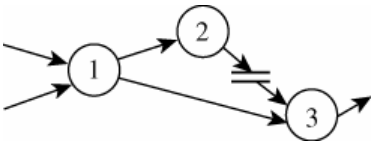
次の図で用いられている表記法はどれか。



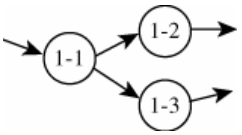
ア　DFD　　イ　状態遷移図　　ウ　流れ図　　エ　ペトリネット

第２種 平成 12 年度春期　問 57

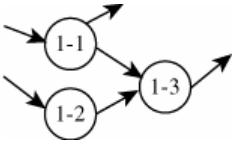
図は、階層化された DFD における、あるレベルの DFD の一部である。その直下のレベルの DFD の記述の仕方として適切なものはどれか。ここで、プロセス n の直下のレベルのプロセスは、プロセス n－1、プロセス n－2、…のように番号をつけるものとする。



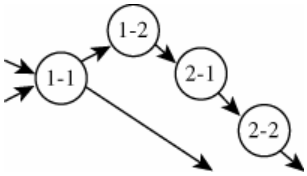
ア



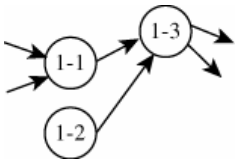
イ



ウ



エ



第１種 平成 12 年度　問 64

受注処理に関する図１の E－R 図に対応した図２の DFD を作成した。図２のデータストア（a）に相当するエンティティは何か。ここで，E－R 図におけるエンティティ間の関係は，矢印のない方を 1，矢印のある方を多とする。

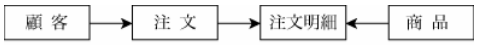


図１　E-R図

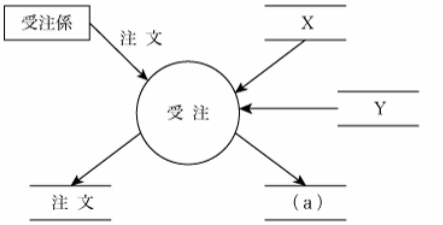


図２　DFD

ア 顧客 イ 在庫 ウ 商品 エ 注文明細　　※平成 14 年問 46 も同一問題。

第２種 平成 9 年度春期　問 60

構造化設計において用いる図で、階層構造の形でプログラムの全体構造を表すものはどれか。

ア　NS チャート　　イ　PERT 図　　ウ　状態遷移図　　エ　バブルチャート

第２種 平成 10 年度春期　問 63

構造化設計の一手法であるバブルチャートの説明として、適切なものはどれか。

ア　階層構造の形で全体構造を表現する。
イ　時間の経過によってどのような状態になるかを表現する。
ウ　処理と処理の間のデータの流れを、データフロー、処理、データストア、外部の 4 種類で表現する。
エ　データの項目と項目の関連を表現する。

基本 平成１３年度春期　問 46

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。

ソフトウェア 平成１３年度　問 48

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。

第２種 平成７年度春期　問 64

外部設計、内部設計、プログラム設計等の工程において、機能分割やプログラム構造の設計を行う際に利用する手法として、適切でないものはどれか。

ア　HIPO　　イ　状態遷移図　　ウ　データフロー図
エ　流れ図　　オ　パレート図

第２種 平成７年度秋期　問 64

構造化分析設計技法に関する用語でないものはどれか。

ア　構造化チャート　　イ　状態遷移図　　ウ　データフローダイアグラム
エ　バブルチャート　　オ　レーダーチャート

物理データ設計

外部設計におけるデータの論理設計の結果をもとにして、物理データ設計を行います。すなわち、データベース、ファイル、メモリにおける、

・データ項目
・データの特性（使用率・増加率）

を検討し、データの物理編成やレイアウトを設計します。

なお、効率だけでなくプログラムの作りやすさも考慮する必要があります。

レコードレイアウトの設計では、

・ キー項目はレベルの高い順に配置する、関連性の高いものはまとめる、項目の属性の同じものはまとめる。
・ 将来の拡張に備えて予備のフィールドを確保する。

などを留意する必要があります。

ドメインとは、データベースにおいて、各列に入っているデータの見出しの役割をしている属性に値として出現することのできるデータの集合である。

第２種 平成７年度秋期　問 63

データ項目名には、ドメイン（定義域）を表現する言葉を含むことが必要である。次のうち、データ項目名として、適切でないものはどれか。

ア　売掛金額　　イ　顧客　　ウ　受注年月　　エ　出荷数量　　オ　商品コード

プログラム設計

プログラム設計では、プログラムの構造化設計を行います。

プログラム設計の手順

以下のような手順で作業をすすめます。

① 内部設計書の確認
② モジュール分割
③ モジュール仕様の作成
④ プログラム設計書の作成
⑤ テストケースの設定
⑥デザインレビュー

モジュール分割技法

規模の大きいプログラムを一まとめのものとして設計すると、処理の流れなどは非常に複雑になってしまいます。そこで、機能単位に分割するのがモジュール化の考え方であり、次のように留意しながら行います。

□１モジュールから呼び出す従属モジュールの数に制限をつける。
□１モジュールが適切な大きさのステップ数になるようにする。
□モジュールからモジュールを呼び出す階層構造があまり深くならないようにする。
□モジュール間のインタフェースが単純になるようにする。

なお、モジュール内の論理を分かりやすくするために、適度にコメントをつけることも重要です。

モジュール分割技法は、大きく２種類に分けることができます。

■データの流れに着目した分割技法
・源泉／分割／吸収分割法（ＳＴＳ）
・トランザクション分割法（ＴＲ法）
・共通分割法

■データ構造に着目した分割技法
・ジャクソン法
・ワーニエ法

■源泉／変換／吸収・分割法（ＳＴＳ分割法）

データは、入力／処理／出力という一連の流れに沿って処理されていくことに着目して、プログラムを以下の三つに分割する技法です。

- **源泉**（*source*）　入力処理機能
- **変換**（*transform*）　データ処理機能
- **吸収**（*sink*）　出力処理機能

分割の手順は、以下のようになります。

①プログラム構造の把握
プログラムの構造を機能中心にとらえ、3〜10 個にまとめます。
②入力データの流れと機能との関連付け
入力データと出力データの主要な流れを、バブルチャートを用い矢印で関連付けます。
③最大抽象点の確定
入力とはいえない点まで抽象化された地点（最大抽象入力点）と、初めての出力データといえる形を表す点（最大抽象出力点）を見つけます。
④直接従属モジュールの定義
上位モジュールから入力・処理・出力に三分割されたモジュールを構造化して、関連付けます。
⑤上位モジュールとのインタフェースの定義
上位モジュールとの情報の受け渡し（モジュール間インタフェース）を定義します。
⑥再分割の可能性のチェック
さらに分割すべきモジュールがあるかどうかをチェックし、あればモジュール分割を繰り返します。

■トランザクション分割法（ＴＲ法）

トランザクション（transaction）すなわち入力データの種類によって、異なる処理を行うときに、トランザクションの種類ごとにモジュール化する分割法です。
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チャートへ変換すします。

■共通機能分割

いくつかのモジュールに共通の機能があるとき、それらの機能を取り出し、別のモジュールとして定義する方法です。

■ジャクソン法

ジャクソン（Jackson）が提唱した方法です。

モジュールの構造を入出力データの構造に対応させて、モジュール設計を行う方法です。データ構造とプログラムのいずれもが、基本・連続・繰返し・選択の三つの基本構造の組合せとして構成されます。

① 入力と出力のデータ構造を定義します。
② 入力・出力データ構造の構成要素間の 1 対 1 の対応関係を見つけます。
③ 出力データ構造をもとに、プログラム構造を作成します。
④ 入力データ構造でプログラム構造の検証を行います。

■ワーニエ法

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す。集合論に基づく設計技法であり、主にファイル処理を中心とする適用業務のモジュール分割に広く用いられています。

① 入力データ構造とプログラム論理構造の対応関係を見つける。
② 部分集合をトップダウン的にブレイクダウンする。
③ フローチャートを作成する。

第２種 平成８年度春期　問 64

“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
オ	変換	吸収	吸収	源泉

第２種 平成７年度春期　問 65

次の記述のうち、データ構造に着目したモジュール分割技法はどれか。

ア　共通機能分割法　　イ　源泉／変換／吸収分割法（STS 法）　ウ　ジャクソン法
エ　シンプレックス法　　オ　トランザクション分割法（TR 法）

第２種 平成９年度春期　問 57

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）

第１種 平成 10 年度　問 62

データの構造に基づいて設計を進める構造化設計技法はどれか。

ア　KJ 法　　イ　源泉／変換／吸収分割法
ウ　ジャクソン法　　エ　トランザクション法

第１種 平成９年度　問 60

ジャクソン法（JSP）に関する記述のうち、正しいものはどれか。

ア　正規化の手法を基にしたデータモデリング手法である。
イ　データフローダイアグラムと E-R 法ダイアグラムを作成する。
ウ　入出力データのデータ構造を連接、選択、繰返しで表す。
エ　プログラムを、データと操作のカプセルとしてとらえる。

第２種 平成 12 年度春期　問 59

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

第２種 平成９年度秋期　問 64

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チャートへ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

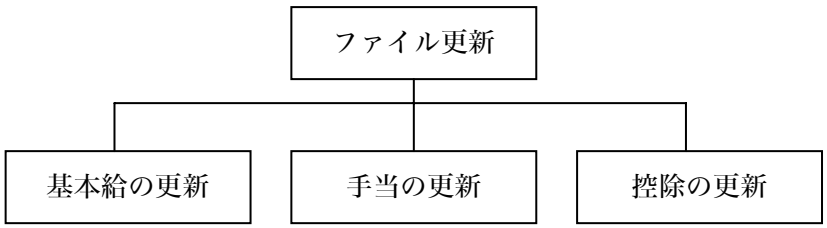
第２種 平成 9 年度秋期　問 63

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、一体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。

第２種 平成 11 年度秋期　問 58

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　　イ　共通機能分割ジャクソン法
ウ　トランザクション分割　　エ　ワーニエ法

第２種 平成 13 年度秋期　問 48

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。

モジュールの独立性

モジュールの設計の際は、以下のことに留意しなければなりません。

□ 1 モジュールから呼び出す従属モジュールの数に制限をつける。
□ 1 モジュールが適切な大きさのステップ数になるようにする。
□ モジュールからモジュールを呼び出す階層構造があまり深くならないようにする。
□ モジュール間のインタフェースが単純になるようにする。

さらに、各モジュールは、可能な限り独立性が高くなければなりません。独立性が高いほど、単純で機能が明確なものとなり、汎用性、信頼性、保守性を向上させることが容易となります。モジュールの独立性の尺度として、モジュール強度とモジュール結合度とがあります。

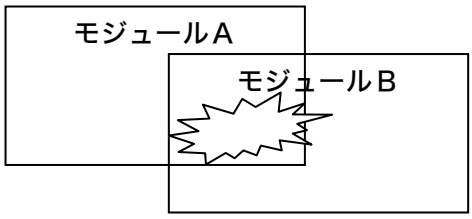
モジュール結合度

他のモジュールとの関連性であり、低いほど独立性が高くなります。以下、結合度の強い（独立性の低い）順に解説します。

① 内容結合

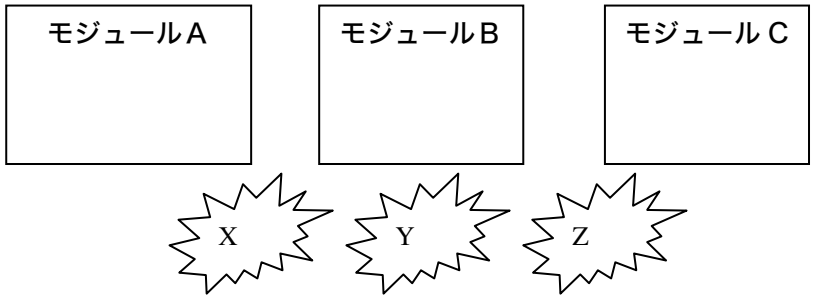
あるモジュールが他のモジュール内のデータを直接参照し、他のモジュールの内容の一部を共有します。

保守が困難である。



② 共通結合

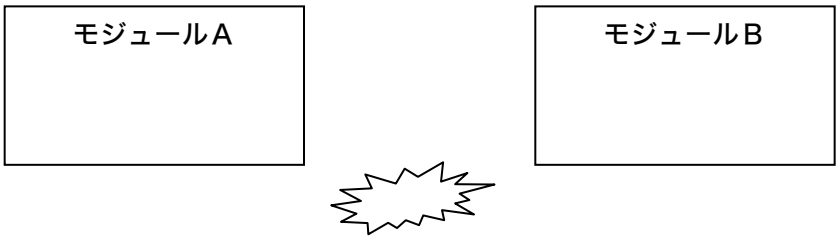
プログラムの共通域に定義したデータを、それに関連するモジュールが共有して参照します（COBOL の GLOBAL, EXTERNAL／Fortran の COMMON など）。



③ 外部結合

外部変数として宣言したデータを、関連するモジュールが共有して参照します。

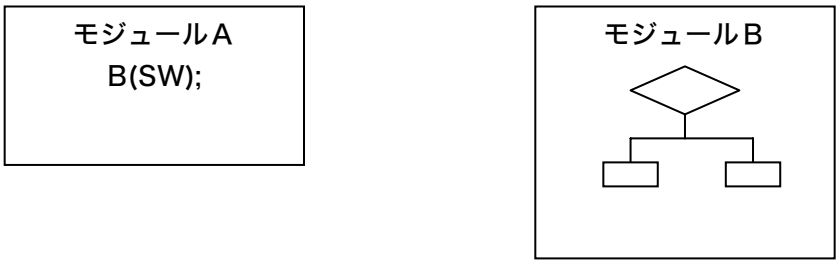
※通常は必要なデータだけを外部宣言する。



④ 制御結合

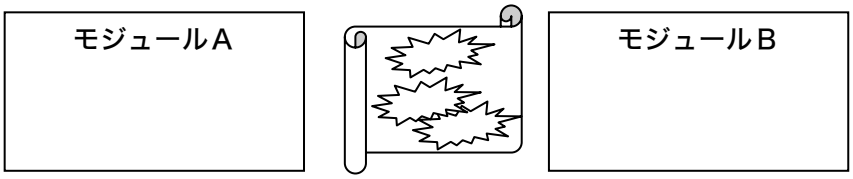
制御情報を、呼び出すモジュールに与え、呼び出すモジュールの実行に影響を与えます。

互いにブラックボックスとして扱えず、論理的強度をもつことになる。



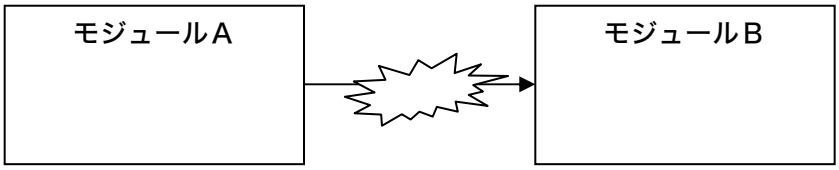
⑤ スタンプ結合

二つのモジュールが同じデータ構造を持った型のデータの受け渡しをします。不必要なデータまで受け渡しする場合があります。



⑥ データ結合

二つのモジュール間で、データ要素のみをパラメタとして受け渡します。

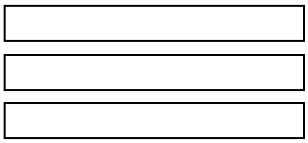


モジュール強度

以下、強度の弱い順に解説します。

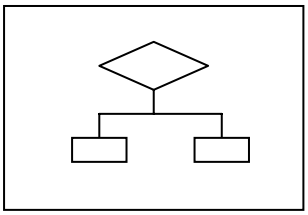
① 暗号的強度

大きさで分割しているため、関連のない複数の機能を含み、特定の機能を定義できないモジュール。



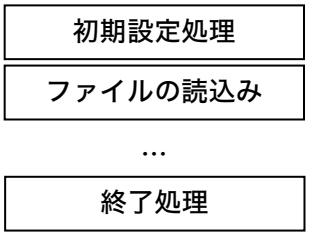
② 論理的強度

関連のある複数の機能を含み、どの機能を実行するかを引数で指定するモジュール。
パラメータが複雑になりやすい。



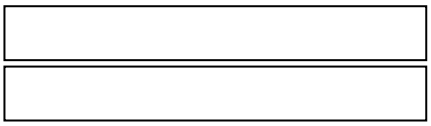
③ 時間的強度

初期設定処理、終了処理など、同時に実行する複数の逐次的処理をまとめたモジュール。
機能間の関連が弱いが、他のモジュールの機能とは強い関連を持つ傾向にある。



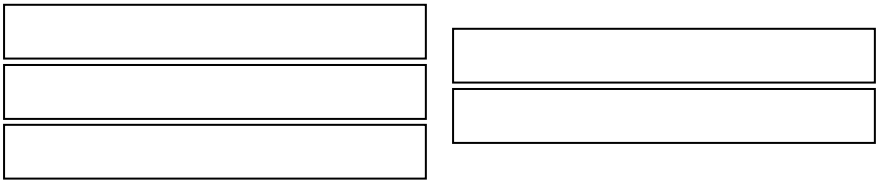
④ 手順的強度

ある手続きを実現するために行う、複数の逐次的処理をまとめたモジュール。たとえば、フローチャートの一部を無作為にモジュール化したような例。



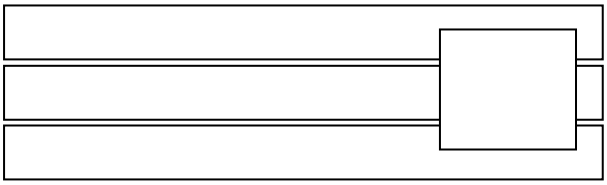
⑤ 連絡的強度

手順的強度の性格をもつことに加えて、同一データを扱うことによって関連性をもったモジュール。



⑥ 情報的強度

特定のデータを扱う複数の機能がまとまっており、機能ごとに別の入り口点をもつモジュール。



⑦ 機能的強度

すべての要素が一つの機能を実行するために関連性をもったモジュール。

第２種 平成 6 年度秋期　問 40

プログラム設計工程でモジュール分割を行なう場合の留意事項の例として、適切でないものはどれか。

ア　1モジュールから呼び出す従属モジュールの数に制限をつける。
イ　1モジュールが適切な大きさのステップ数になるようにする。
ウ　モジュールからモジュールを呼び出す階層構造があまり深くならないようにする。
エ　モジュール間のインターフェースが単純になるようにする。
オ　モジュール内の論理を分かりやすくするために、適度にコメントをつける。

第２種 平成 8 年度春期　問 65

次のうち、モジュールの結合度が最小であるといわれている方式はどれか。

ア　外部結合　　イ　共通結合　　ウ　スタンプ結合　　エ　制御結合
オ　データ結合

第２種 平成 11 年度秋期　問 57

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

第２種 平成 9 年度春期　問 58

モジュール間の情報の受渡しがパラメタだけで行われ、結合度が最も弱いモジュール結合はどれか。

ア　外部結合　　イ　共通結合　　ウ　制御結合　　エ　データ結合

第２種 平成 11 年度春期　問 58

モジュール間の情報の受渡しがパラメタだけで行われ、結合度が最も弱いモジュール結合はどれか。

ア　外部結合　　イ　共通結合　　ウ　制御結合　　エ　データ結合

基本 平成 13 年度秋期　問 49

トップダウンアプローチによって、プログラムが階層構造になるように構造化設計を行い N 個のモジュールに分割した。このプログラムのモジュール間インタフェースの個数を表す式はどれか。ここで、下位のモジュールは上位のモジュールのどれか一つとだけインタフェースをもつものとする。

ア　N　　イ　N^2　　ウ　N－1　　エ　N (N－1) ／2

ソフト 平成 14 年度　問 50

モジュールの独立性を高めるためには、モジュールの結合度を弱くする必要がある。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、モジュール間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。

第1種 平成 11 年度　問 63

構造化設計から得られるモジュール分割の良否を、モジュール結合度の視点から判断するときの記述として、適切なものはどれか。

ア　ソフトウェア全体のモジュール分割の良否は、モジュール間の結合度のうちで最も弱いものがどのレベルにあるかで判断するのが望ましい。
イ　データ領域は、すべてのモジュールからアクセスできるようになっていることが望ましい。
ウ　モジュール構造図で遠くにあるモジュール間は、共通のデータ領域を介して情報を交換する方式が最も望ましい。
エ　呼び出す側と呼び出される側のモジュール間はすべて、データ項目だけを引数で受渡しするのが最も望ましい。

ソフト 平成 13 年度　問 51

モジュール強度のレベルの一つとして、情報的強度がある。情報的強度を持つモジュールの例として、適切なものはどれか。

ア　関連した複数の機能を一つにまとめており、どの機能が実行されるかは、呼び出されるときの引数の値によって決定されるモジュール。
イ　データとデータの操作を独立のものとして取り扱うことが可能なので、サブシステムの独立性を高めることができる。
ウ　データの詳細な構造について知らないとアクセスでないので、データのセキュリティが強くなる。
エ　データを制御する手続きは一意に定義できないが、データ構造の一貫性は維持される。

第1種 平成 10 年度　問 63

次のように設計・コーディングしたモジュールのうち、モジュール強度が最も高いものはどれか。

ア　ある木構造データを扱う機能をデータとともに一つにまとめ、木構造データをモジュールの外から見えないようにした。
イ　初期設定の操作が複数の機能のそれぞれに必要であるが、ある時点で一括して実行できると考えられたので、一つのモジュールにまとめてコーディングした。
ウ　ニつの機能 A、B のコードには重複する部分が多いので、A、B を一つのモジュールとしてコーディングし、A、B の機能を使い分けるための引数を設けた。
エ　ニつの機能 A、B は必ず A、B の順番に実行され、しかも A で計算した結果を B で使うことがあるので、一つのモジュールにまとめてコーディングした。

テスト

プログラムがいったん完成すると、システムとして、要求仕様通りのものであるかどうかを確認し、バグを見つけるために、段階的なテストが必要です。

テストの手順

以下のように、より小さい単位から次第に大きい単位へという手順でテスト作業を進めます。

単体テスト → 結合テスト → システムテスト（総合テスト） → 運用テスト

① 単体テスト

モジュールを結合してしまった後では、テストに労力がかかりますので、プログラミングの段階で各モジュール単位でのテストを行います。

ブラックボックステストを中心に、ホワイトボックステストなどを行います。

② 結合テスト

モジュールを結合して一つのプログラムとして作動するかどうかのテストであり、モジュール間のインタフェースを検証するために、プログラムの入出力を含む一貫テストを行います。

たとえ単体テストでエラーが発見されなくても、結合テストでエラーが見つかることもあります。

③ システムテスト（総合テスト）

プログラム全体をとおして、システムとして作動するかテストします。

例：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。
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。
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。
他のシステムとのインタフェーステストを行う。

④ 運用テスト

実際のデータを使って、利用者も含めてシステムが実務上運用できるようにテストします。

基本 平成 13 年度秋期　問 51

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

第２種 平成 7 年度春期　問 68

システム開発におけるテスト工程の正しい順序はどれか。

ア　運用テスト　→　システムテスト　→　単体テスト　→　結合テスト
イ　運用テスト　→　単体テスト　→　結合テスト　→　システムテスト
ウ　結合テスト　→　システムテスト　→　単体テスト　→　運用テスト
エ　単体テスト　→　結合テスト　→　運用テスト　→　システムテスト
オ　単体テスト　→　結合テスト　→　システムテスト　→　運用テスト

第２種 平成 10 年度秋期　問 62

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

単体テスト

単体テストは、プログラミングの段階において各モジュール単位で行います。
ブラックボックステスト・ホワイトボックステストなどがあります。

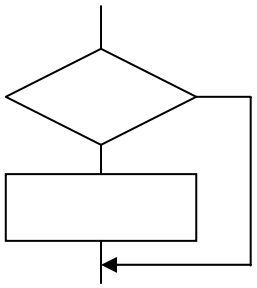
ホワイトボックステスト

プログラムの内部構造やアルゴリズムに着目してロジックを調べるテスト方法です。テストケースの設計法として、条件網羅、命令網羅、判定条件網羅、判定条件／条件網羅、複数条件網羅などの手法を使います。

■ 命令網羅

プログラム中の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。

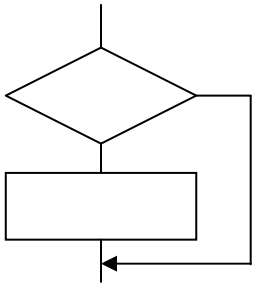
```
if (x != 0)
    z = a;
```



■ 判定条件網羅

判定条件で、真、偽ともに少なくとも 1 回は実行するようにテストケースを設計する。

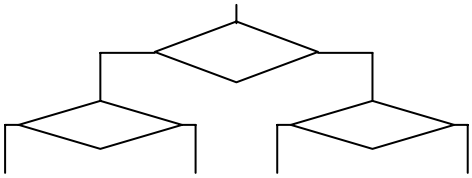
```
if (x != 0)
    z = a;
```



■ 条件網羅

判定条件の真と偽について、それぞれの組合せを満たし、かつ少なくとも 1 回は実行するようにテストケースを設計する。

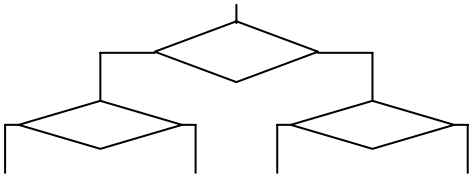
```
if (x != 0 || y == 5)
    z = a;
```



■ 複数条件網羅

判定条件の真と偽について、あらゆる組合せの経路を網羅し、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。

```
if (x != 0 || y == 5)
    z = a;
```



第２種 平成 6 年度秋期　問 43

プログラムの内部構造やアルゴリズムに着目して行うテストを表す用語として、最も適切なものはどれか。

ア　システムテスト　　イ　トップダウンテスト　ウ　ブラックボックステスト
エ　ホワイトボックステスト　オ　ボトムアップテスト

第２種 平成 7 年度秋期　問 60

ホワイトボックステストで用いられる手法を二つ選べ。

ア　因果グラフ（原因結果グラフ）　イ　限界値分析　ウ　条件網羅
エ　同値分割　オ　命令網羅

第２種 平成 8 年度秋期　問 64

ホワイトボックス法に関するテストケースの作成方法はどれか。

ア　エラー推測　イ　原因－結果グラフ　ウ　限界値分析　エ　条件網羅
オ　同値分析

第２種 平成 11 年度春期　問 66

ホワイトボックステストで利用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　実験計画法
ウ　条件網羅　エ　同値分割

第２種 平成 10 年度秋期　問 64

プログラムのテスト手法の一つであるホワイトボックステストの説明として、適切なものはどれか。

ア　最初にプログラム構成の最下位モジュールをテストする。次に、それを呼び出すモジュールを結合してテストする。このように、次々と上位に広げていくテストである。
イ　最初にプログラム構成の最上位モジュールをテストする。次に、この上位モジュールから呼ばれる下位モジュールを結合してテストする。このように、次々と下位に広げていくテストである。
ウ　プログラムの外部仕様に着目し、入力の可能性がある値のすべての組合せをテストする手法である。同値分割、限界値分析、原因結果グラフなどの技法がある。
エ　プログラムの内部構造に着目し、ロジックを調べ、すべての経路が実行されるようにテストする手法である。命令網羅、条件網羅などの技法がある。

第２種 平成 12 年度春期　問 62

テスト手法の一つであるホワイトボックス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、順次結合してテストする。
イ　上位のモジュールから下位のモジュールへと、順次結合してテストする。
ウ　モジュールの内部構造に注目して、テストする。
エ　モジュールの内部構造を考慮することなく、仕様書どおり機能が作動するかどうかをテストする。

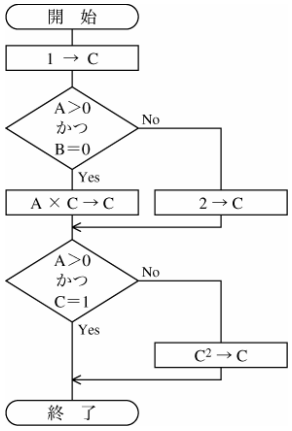
第１種 平成 13 年度 問 52

ホワイトボックステストのテストデータ作成に関する記述として、適切なものはどれか。

ア　入力データを同値分割法に基づいて分析し、テストデータを作成する。
イ　プログラムのアルゴリズムなど、内部構造に基づいてテストデータを作成する。
ウ　プログラムの機能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に基づいてテストデータを作成する。

第１種 平成９年度　問 64

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

ブラックボックステスト

プログラムの設計書からその機能を中心にテストする方法で適切な入力を行い、期待した通りの出力が得られるかを、内部構造を考慮することなくテストする手法です。因果グラフ、限界値分析、同値分割などの手法を使います。

■ 同値分割

同じ特性をもついくつかのグループ（同値クラス）に分割して、一つのデータを代表値としてそれぞれのグループを分割する方法。

有効同値クラス：正当な範囲にあるデータクラス

無効同値クラス：不当な範囲にあるデータクラス

■ 限界値分析

有効同値クラスの境界値をテストデータとして分析する方法。

例：最大値と最小値および、それらを一つ超えた値。

■ 因果グラフ

クラス分けが困難な場合に有効なテストケースの設計方法。

①仕様書をもとに原因（入力）と結果（出力）を洗い出す。

②原因（入力）と結果（出力）のグラフを作成し、論理的な関係を明確にする。

③作成したグラフをもとにデシジョンテーブルを作成し、それをもとにテストデータを作成する。

■ 実験計画法

考えられるすべてのテストケースを設計することは困難であり、大量のデータをテストするための統計的な分析法テストデータを分析し、その結果をテストケースとして用いる。

第２種 平成 10 年度春期　問 66

モジュールの内部構造を考慮することなく、仕様書どおりに機能するかどうか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

第２種 平成 11 年度春期　問 63

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

第２種 平成 10 年度春期　問 64

テスト手法の一つである限界値分析におけるテストデータとして、適切なものはどれか。

ア　最小値と最大値
イ　最小値と最大値、及びそれらを一つ超えた値
ウ　最小値とそれを一つ超えた値
エ　最大値とそれを一つ超えた値

第１種 平成 10 年度 問 65

ブラックボックス法によるテストケースの設計に関する記述として、適切なものはどれか。

ア　実データから無作為にテストデータを抽出する。
イ　プログラムの外部仕様の観点からテストケースを設計する。
ウ　プログラムのすべての命令が少なくとも 1 回は実行されるようにテストケースを設計する。
エ　プログラムの内部ロジックに基づいてテストケースを設計する。

第１種 平成 12 年度 問 65

ブラックボックス法によるテストケースの設計に関する記述として、適切なものはどれか。

ア　実データから無作為にテストデータを抽出する。
イ　プログラムの外部仕様の観点からテストケースを設計する。
ウ　プログラムのすべての命令が少なくとも 1 回は実行されるようにテストケースを設計する。
エ　プログラムの内部ロジックに基づいてテストケースを設計する。

第１種 平成 10 年度 問 66

帳票 1 ページごとに、ヘッダと最大 30 件分のレコードを出力するプログラムがある。出力するレコード件数について限界値分析法を用いる場合、テストケースとして設定する入力レコード件数の組合せのうち，最も適切なものはどれか。

ア　0，1，29，30，31，59，60，61
イ　0，10，20，30，40，50，60，70
ウ　0，30，60，90，120，150，180，210
エ　0，30，300，3000，16383，16384，32767，32768

第１種 平成 11 年度 問 65

入力データとして 100～200 の範囲の整数を受け付けるプログラムのテストケースを、限界値分析法によって設計する。同値クラスとして 99 以下、100～200、201 以上の三つを設定したときのテストデータの組合せとして、最も適切なものはどれか。

ア　100 200
イ　50 150 250
ウ　99 100 200 201
エ　50 100 150 200 250

結合テスト

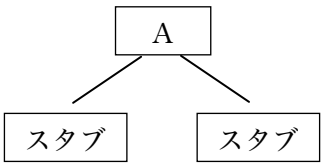
モジュールを結合して一つのプログラムとして作動するかどうかのテストであり、モジュール間のインタフェースを検証するために、プログラムの入出力を含む一貫テストを行います。

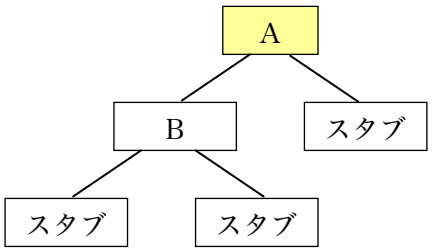
たとえ単体テストでエラーが発見されなくても、結合テストでエラーが見つかることもあります。

トップダウンテスト

結合テストの方式であり、最上位モジュールから順にモジュールテストを行う方式です。モジュールのテストの際、下位モジュールが未完成の場合、**スタブ**と呼ばれるテストモジュールを用意します。

新規のシステム開発に適しています。

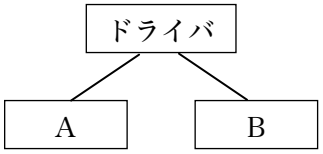


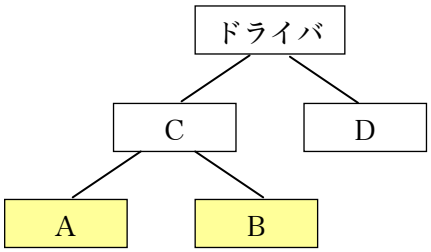


ボトムアップテスト

最下位モジュールから順にモジュールテストを行う方式です。下位モジュールをテストする際、その上位レベルのモジュールが未完成の場合、**ドライバ**と呼ばれるテストモジュールを用意します。

既存のシステムに修正を加えて開発するような場合に適しています。

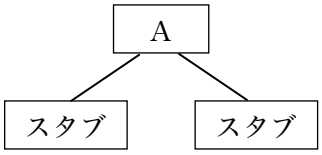


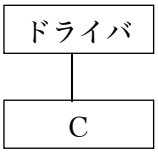


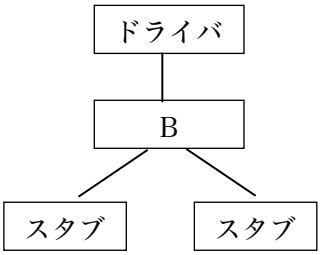
ビックバンテスト

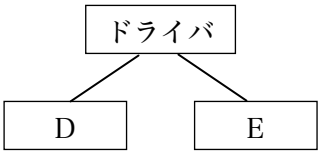
モジュールの単体テストがすべて終了してから、それらのモジュールを一度に結合してテストする方法です。以下のような特徴があります。

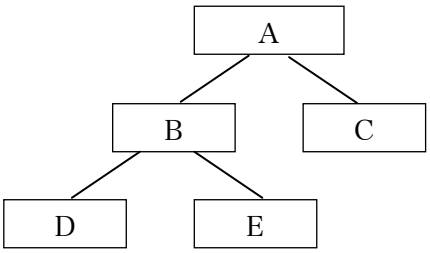
・単体テストが済んだ後なので、信頼性が高い。
・結合テストの際に、スタブやドライバが必要となる。
・すべてのモジュールを最初からテストできる。
・単体テストがすべて終了しなければ、テストを実施することができない。
・モジュール間インタフェースのエラーを早期に発見することが困難である。
・エラー発見後のデバッグが面倒である。











第２種 平成６年度秋期　問 42

システム開発のテスト工程において、単体テスト（モジュールテスト）の直後に行う結合テストに関する記述として、適切なものはどれか。

ア　システムが外部仕様書に示された機能どおりに実現されているかを検証するテストである。
イ　処理時間や応答時間の目標が達成されているかを検証するテストである。
ウ　接続する入出力機器や通信機器の種類と数に問題がないかを検証するテストである。
エ　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テストである。
オ　目標どおりのジョブの多重度走行や端末の同時接続が実現できるかどうかを検証するテストである。

第２種 平成９年度春期　問 63

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。

第２種 平成 11 年度春期　問 62

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

第２種 平成 12 年度秋期　問 61

システムの開発工程の段階を要求定義、外部設計、内部設計、プログラム設計及びプログラミングに分けた場合、外部設計の検証を主目的として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

第２種 平成７年度秋期　問 59

トップダウンテストで、テスト対象のモジュールから呼び出され、下位モジュールの代わりをつとめるものはどれか。

ア　インスタンス　　イ　エンティティ　　ウ　スタブ　　エ　デーモン　　オ　ドライバ

第２種 平成９年度春期　問 62

トップダウンテストにおいて、被テストモジュールの下位モジュールの機能を代行するものはどれか。

ア　カプセル　　イ　スタブ　　ウ　ダミー　　エ　ドライバ

第２種 平成 11 年度秋期　問 63

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。

第２種 平成 6 年度秋期　問 44

プログラムのテストに利用されるスタブに関して、適切な記述はどれか。

ア　開発の各段階で複数の関係者が集まって、幾つかの異なった角度から机上での検査を行い、プログラムの欠陥や誤りなどの問題点を検出し、訂正する。

イ　指定した特定の命令が実行されるたびに、レジスタや主記憶装置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。

ウ　トップダウン的にプログラムのテストを行うとき、既に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。

エ　プログラムの実行中、適宜変数やレジスタ等の内容を検査し、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。

オ　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。

第２種 平成 11 年度春期　問 64

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。

第２種 平成 11 年度秋期　問 64

結合テストで用いられるスタブの役割の記述として、適切なものはどれか。

ア　テスト完了のモジュールの代わりに結合される。

イ　テスト対象のモジュールからの呼出し命令の条件に合わせて、値を返す。

ウ　テスト対象のモジュールからの呼出し命令の条件に合わせて、テストデータを自動生成する。

エ　テスト対象のモジュールを呼出し命令で呼び出す。

第２種 平成 10 年度秋期　問 63

ボトムアップテストに関する記述として、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のもとで行うテストである。

エ　ドライバが必要である。

第２種 平成 7 年度春期　問 70

ボトムアップテストに関する記述として、最も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 上位のモジュールがそろってからテストを開始する。

ウ スタブを作る必要がある。

エ 既に稼働しているモジュールのもとで行うテストである。

オ ドライバを作る必要がある。

第２種 平成 9 年度秋期　問 67

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　下流のモジュールから上流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。

第２種 平成 8 年度秋期　問 63

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モジュールの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

オ　リンケージエディタ

ソフトウェア 平成 14 年度　問 52

トップダウンテストと比較したときのボトムアップテストに関する記述のうち、適切なものはどれか。

ア　既存のシステムを修正してシステム開発する場合よりも、新規に開発するシステムに適用すると効果がある。

イ　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発見されたときの影響が大きい。
ウ　未開発の上位のモジュールを代行するスタブが必要である。
エ　モジュール数の多い下位の部分から開発していくことになるので、開発の初期の段階ではプログラミングとテストの並行作業が困難である。

第２種 平成 8 年度春期　問 62

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となる。
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はない。
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すべ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。
オ　ループ処理ロジックのテストでは、最も多く発生するループ回数についてテストすればよい。

第２種 平成 8 年度春期　問 63

結合テストの 1 方法であるビックバンテストを説明している記述はどれか。

ア　テストの開始前に机上の検証で、誤りを発見して修正する方法。
イ　モジュール構成図の最下位モジュールを最初にテストし、順次上位モジュールを結合してテストする方法。
ウ　モジュール構成図の最上位モジュールを最初にテストし、順次下位モジュールを結合してテストする方法。
エ　モジュールの単体テストがすべて終了してから、それらのモジュールを一度に結合してテストする方法。
オ　モジュールの入力と出力のパターンからテストケースを決定し、テストする方法。

第２種 平成 9 年度春期　問 64

ビッグバンテストの手法の説明として、適切なものはどれか。

ア　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テストする。
イ　モジュールの内部構造に着目してテストする。
ウ　モジュールの内部構造を考慮することなく、仕様書どおりに機能が作動するどうかをテストする。
エ　モジュールを一度に結合してテストする。

テストデータ

テストを行う際は、事前にテストケースを設定し、その入力データに対する結果を準備しておく必要があります。

第２種 平成７年度春期　問 69

プログラムのテストに関する次の記述のうち、最も適切なものはどれか。

ア　テストデータは、正しい入力データについてだけ作成する。
イ　テスト前に、テストの入力データに対する正しい結果を準備しておく。
ウ　プログラムが複数のモジュールに分かれているときは、まずモジュールの結合テストから行う。
エ　プログラム中のすべての分岐点から、任意に選んだ数個についてテストする。
オ　プログラムを作成した本人だけがテストする。

第２種 平成７年度秋期　問 61

プログラムの検査に用いるテストデータに関して、適切な記述はどれか。

ア　事前にテストケースを設定し、それに沿ったテストデータを準備する。
イ　正しく処理されるテストデータだけを、テストの進行に応じて準備する。
ウ　テスト用のデータとして、運用時に処理されるデータの量の 2 割程度を用意する。
エ　入力段階のチェックでエラーになるテストデータは、準備する必要がない。
オ　ブラックボックステストでは、プログラムの論理を分析したうえでテストデータを作成する。

第２種 平成 10 年度春期　問 65

プログラムの検査に用いるテストデータに関して、適切な記述はどれか。

ア　事前にテストケースを設定し、それに沿ったテストデータを準備する。
イ　正しく処理されるテストデータだけを、テストの進行に応じて準備する。
ウ　テスト用のデータとして、運用時に処理されるデータ量の 2 割程度を用意する。
エ　入力段階のチェックでエラーになるテストデータは、準備する必要がない。

運用テスト

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、最終テストです。ユーザ部門が自らテストケースを設定し、実際に運用するときと同じ条件でテストを行い、システムが要求仕様を満たしているかどうかを検証します。開発したシステムをユーザ部門に受け入れてもらうためのテストなので、承認テスト、または受け入れテストともいわれます。

実際に業務で利用しているコンピュータ上で作業することになりますから、業務に支障のないようにテストを行わなければなりません。

第２種 平成 12 年度春期　問 63

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。

レグレッションテスト

ソフトウェ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テスト手法のことで、回帰テスト、退行テストとも呼ばれます。
※プログラムのバグを直したつもりが、別の箇所に不具合がでる、といったことはよくありますね。

第２種 平成 11 年度秋期　問 61

ソフトウェアのテスト方法のうち、ソフトウェ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　レグレッションテスト

基本 平成 13 年度春期　問 49

ソフトウェアのテスト方法のうち、ソフトウェア保守のために変更した箇所が、ほかの部分に影響してい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　レグレッションテスト

ソフトウェアの保守費用について

ソフトウェアを利用するにしたがって、「すりへる」とか「キレが悪くなる」などということはありませんから、“保守”は必要ないように思われますが、現実はそうではありません。ある統計によると、ソフトウェアにかかる費用の 70%が保守に費やされています。

本来の保守 ‥‥ 仕様変更などによる修正等
怪しい保守 ‥‥ 時期遅れのデバッグ

保守のコストの割合*

ユーザ要求の変更	41.8%
データフォーマットの変更	17.4%
緊急の修正	12.4%
定期的なデバッグ	9.0%
ハードウェアの変更	6.2%
ドキュメント作業	5.5%
効率性向上のための修正	4.0%
その他	3.4%

変更を受け入れにくく、データの物理構造に依存するソフトウェアの存在が、保守を妨げることになっていることに気付きましょう。

たとえば、東京 23 区内の市内局番が 3 桁と思いこんでいたソフトウェアは、4 桁への変更が行われた際に、大きなデータフォーマットの変更を強いられたはずです。

ユーザ要求の変更が５分の２以上をしめることは、ソフトウェアの「拡張性」が欠けているからとも考えられます。

緊急の修正が、定期的なデバッグより大きいというのも注目に値します。大きなプレッシャーのもとでの作業が要求されます。

> ■実例■　私は学生時代に“ 自動車ローン（自動車を担保にした「サラ金」みたいなもの）” の管理システムのファイル修復作業を行ったことがあります。受付に座っている美人のお姉さんの後ろには鏡。もちろん、ふつうの鏡ではなくてマジックミラー。怪しげな、その手のおじさん達に、マジックミラーごしに覗かれながらの作業には、ものすごいプレッシャーがかかったものです。

もちろん、緊急の修正によって、新たなバグが発生することも考えられます。

* B.P.Lients and E.B.Swanson、 Softre Maintenance: A User/Management Tug of War、 Data Management、 pp.26-30、 Apr. 1979

オブジェクト指向

オブジェクト指向の目的は、ソフトウェアの生産性を向上させることです。オブジェクト指向型のプログラミング言語としては、Smalltalk、C++、Java があります。

オブジェクト・クラス

たとえば、個々の車を**オブジェクト**と考えましょう。車の一般的な性質を**クラス**といいます。車クラスには、車長、車幅、排気量といった**データ**（属性）と、走る、とまるといった振舞い（**メソッド**）から構成されます。

カプセル化

データとメソッドを一つにまとめ、オブジェクトの実装の詳細をユーザから見えなくして外部から直接アクセスできないようにすることを**カプセル化**といいます。

内部データ構造やメソッドの実装を変更しても、その影響をほかのオブジェクトに及ぼしにくくなります。

抽象化

カプセル化を行って、クラスを定義すること。

インスタンス

クラスに基づく実体（いわゆる“変数”に相当）が**インスタンス**です。

※抽象クラスのインスタンスを生成することはできません。

メッセージ

車のオブジェクトに対して、「走れ。」「とまれ。」といった依頼が**メッセージ**です。オブジェクト間の情報交換の手段です。

継承（インヘリタンス）

既存のクラスの特性をもとに、新しいクラスを定義すること。元のクラスを**スーパークラス（基本クラス）**、新しく作られたクラスを**サブクラス（派生クラス）**と呼ぶ。サブクラスは、スーパークラスのデータやメソッドを継承することになる。

基本クラスに対して、サブクラスを作ることによって、基本クラスのデータや機能を再利用する、**開かれた（ホワイトボックス型）再利用**では、基本クラスで定義したデータや機能に対して差分を記述すればよく、開発効率が高くなります。

クラス階層を導く手法としては、汎化／専化、集約化／分解、グループ化などの抽象化操作があります。

多相性／多様性／多態性（ポリモーフィズム）

複数のサブクラスに対して、同一名のメッセージを送信した場合、それぞれのクラスが、固有の振る舞いを実行できること。

委譲（デリゲーション）

あるオブジェクトに操作を適用したとき、関連するオブジェクトに対してもその操作が自動的に適用される仕組み。

第２種 平成８年度秋期　問 59

オブジェクト指向の特徴として、適切でないものはどれか。

ア　カプセル化　　イ　継承　　ウ　多相性　　エ　抽象化　　オ　データの正規化

第２種 平成９年度春期　問 56

オブジェクト指向に関する記述として、適切なものはどれか。

ア　オブジェクト間の情報交換は、インスタンスで行う。
イ　オブジェクトとは、クラスの性質を記述したものである。
ウ　カプセル化とは、クラスをライブラリ化することである。
エ　クラスは、その親のクラスからメソッドを継承できる。

第２種 平成９年度秋期　問 59

オブジェクト指向プログラムにおいて、データとメソッドを一つにまとめ、オブジェクトの実装の詳細をユーザから見えなくすることを何というか。

ア　インスタンス　　イ　カプセル化　　ウ　クラスタ化　　エ　抽象化

第２種 平成 12 年度春期　問 58

オブジェクト指向プログラムにおいて、データとメソッドを一つにまとめ、オブジェクトの実装の詳細をユーザから見えなくすることを何と呼ぶか。

ア　インスタンス　　イ　カプセル化　　ウ　クラスタ化　　エ　抽象化

第２種 平成 12 年度秋期　問 53

オブジェクト指向に関する記述として、適切なものはどれか。

ア　オブジェクトは、クラスのテンプレートである。
イ　カプセル化とは、クラスをライブラリ化することである。
ウ　クラスは、必ず一つ以上のインスタンスをもつ。
エ　クラスは、その親のクラスから属性とメソッドを継承できる。

基本 平成 13 年度春期　問 47

オブジェクト指向に関する記述のうち、適切なものはどれか。

ア　オブジェクト指向モデルでは、抽象化の対象となるオブジェクトの操作をあらかじめ指定しなければならない。
イ　カプセル化によって、オブジェクト間の相互依存性を高めることができる。
ウ　クラスの変更を行う場合には、そのクラスの上位にあるすべてのクラスの変更が必要となる。
エ　継承という概念によって、モデルの拡張や変更の際に変更部分を局所化できる。

第１種 平成９年度　問 42

オブジェクト指向におけるインヘリタンスに関する記述として、正しいものはどれか。

ア　あるクラスの下にサブクラスを定義するとき、上のクラスで定義されたメソッドとデータをサブクラスで引き継いで使うことができる。
イ　オブジェクトの性格を決めるデータ構造や値を隠ぺいし、オブジェクトを外部から直接アクセスすることを禁止する。
ウ　オブジェクトのデータ構造や処理方法を変更した場合、外部への影響を避けることができ、オブジェクトの独立性を向上させることができる。
エ　同一のデータ構造と同一の手続きをもつオブジェクトをまとめて表現したものである。

第１種 平成 10 年度　問 61

オブジェクト指向モデルに関する記述のうち、正しいものはどれか。

ア　共通の属性をもつオブジェクトは、一つのメソッドとして定義される。
イ　クラス階層の間で、上位のクラスの属性を下位のクラスに引き継ぐことを、メッセージパッシングという。
ウ　クラス階層を導く手法として、汎化／専化、集約化／分解、グループ化などの抽象化操作がある。
エ　データ、そのデータを操作するプロセス及び制約をオブジェクト内に封じ込めることを、集約化という。

第１種 平成 11 年度　問 62

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きを一つにして、オブジェクトの内部に隠ぺいすること

第１種 平成 12 年度　問 62

オブジェクト指向でシステムを開発する場合、カプセル化の効果として適切なものはどれか。
ア　オブジェクトの内部データ構造やメソッドの実装を変更しても、その影響をほかのオブジェクトに及ぼしにくい。
イ　親クラスの属性を子クラスが利用できるので、親クラスの属性を子クラスの属性の定義に利用できる。
ウ　既存の型に加えてユーザ定義型を追加できるので、問題領域に合わせてプログラムの仕様を拡張できる。
エ　同一メッセージを各オブジェクトに送っても、オブジェクトによって動作が異なるので、メッセージを受け取るオブジェクトの種類が増えても、メッセージを送るオブジェクトには影響がない。

ソフト 平成 14 年度　問 48

オブジェクト指向における委譲に関する説明として、適切なものはどれか。

ア　あるオブジェクトが複数のオブジェクトから構成される仕組み
イ　あるオブジェクトに操作を適用したとき、関連するオブジェクトに対してもその操作が自動的に適用される仕組み
ウ　あるオブジェクトに対する操作を、その内部でほかのオブジェクトに依頼する仕組み
エ　下位のクラスが上位のクラスの属性や操作を引き継ぐ仕組み

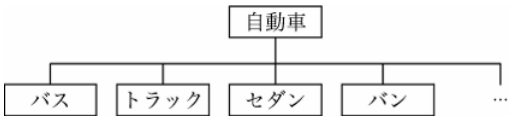
ネットワークスペシャリスト 平成 12 年度　問 70

オブジェクト指向開発において、オブジェクトのもつ振る舞いを記述したものを何というか。

ア　インスタンス　　イ　クラス
ウ　メソッド　　エ　メッセージ

上級システムアドミニストレータ 平成 12 年度　問 29

オブジェクト指向において、図のような階層のクラスを構成する場合、クラス間の関係に関する説明のうち、適切なものはどれか。



ア　“自動車”のデータを“バス”、“トラック”などのクラスで引き継ぐことをインスタンスという。
イ　“自動車”は、“バス”、“トラック”などのクラスから見て、サブクラスといえる。
ウ　“バス”、“トラック”などのクラスの共通部分を抽出して、“自動車”のクラスとして定義することを汎化という。
エ　“バス”、“トラック”などのそれぞれのクラスの違いを、“自動車”のクラスとして定義することを特化という。

プロジェクトマネージャ 平成 12 年度　問 45

オブジェクト指向でシステムを開発する場合、カプセル化の効果として適切なものはどれか。
ア　オブジェクトの内部データ構造やメソッドの実装を変更しても、その影響をほかのオブジェクトに及ぼしにくい。
イ　親クラスの属性を子クラスが利用できるので、親クラスの属性を子クラスの属性の定義に利用できる。
ウ　既存の型に加えてユーザ定義型を追加できるので、問題領域に合わせてプログラムの仕様を拡張できる。
エ　同一メッセージを各オブジェクトに送っても、オブジェクトによって動作が異なるので、メッセージを受け取るオブジェクトの種類が増えても、メッセージを送るオブジェクトには影響がない。

プロジェクトマネージャ 平成 12 年度　問 46

オブジェクト指向の概念において、（私の父：公務員）が“私の父は公務員”というインスタンスとクラスの関係を意味するとしたとき、同じ関係となるものはどれか。

ア　（赤い車：乗り物）　　イ　（オーストラリア：国）
ウ　（私の父：私の母）　　エ　（私の部屋：私の家）

システムアナリスト 平成 12 年度　問 42

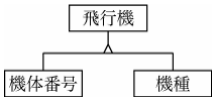
オブジェクト指向における、開かれた（ホワイトボックス型）再利用とは、基本クラスに対して、サブクラスを作ることによって、基本クラスのデータや機能を再利用することである。この方式のオブジェクト指向の再利用技術に関する記述のうち、適切なものはどれか。

ア　基本クラスで定義したデータが保護されるので、安全性の高いプログラムが開発できる。
イ　基本クラスで定義したデータや機能に対して差分を記述すればよく、開発効率が高い。
ウ　基本クラスの変更は、サブクラスに影響しない。
エ　基本クラスを複数のアプリケーション開発に利用することができるが、そのサブクラスを再利用することはできない。

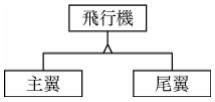
アプリケーションエンジニアリング 平成 12 年度　問 28

汎化（is-a 関係）を表す図はどれか。

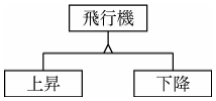
ア



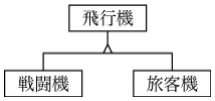
イ



ウ

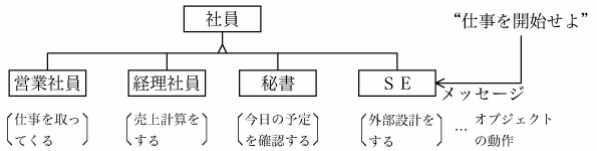


エ



アプリケーションエンジニアリング 平成 12 年度　問 47

オブジェクト指向において，図のようなクラス構造のオブジェクトを考えた場合、“仕事を開始せよ”というメッセージに対して営業社員や経理社員といったオブジェクトがそれぞれの動作をすることを何というか。



ア　インヘリタンス　　イ　カプセル化
ウ　情報隠ぺい　　エ　ポリモルフィズム

参考文献

1) Bertrand Meyer（二木厚吉監訳）『オブジェクト指向入門』，アスキー，1990
2) 川村一樹『ソフトウェア工学入門（第２版)』，啓学出版，1992
3) 石井康雄『ソフトウェア工学入門』、日科技連，1989
4) 藤本喜弘ら『第二種情報処理技術者試験過去問題 ＆分析'98 春期・秋期試験対応版』，経林書房，1997
5) 加藤昭『第２種【要点・重点】短期集中速攻対策』，技術評論社，1998
6) 財団法人日本情報処理開発協会 中央情報教育研究所『第二種 共通テキスト③ システム開発』，コンピュータエージ，1998
7) オーム社編『2003 年度版 ソフトウェア開発試験』，オーム，2002
8) 河村知信『第一種情報処理技術者過去問題＆分析 2000 年度版』，経林書房，1999