JaSST'09 Kansai

『ソフトウェア品質検証、評価技術の勘所』

2009/07/31

NARAコンサルティング
奈良隆正

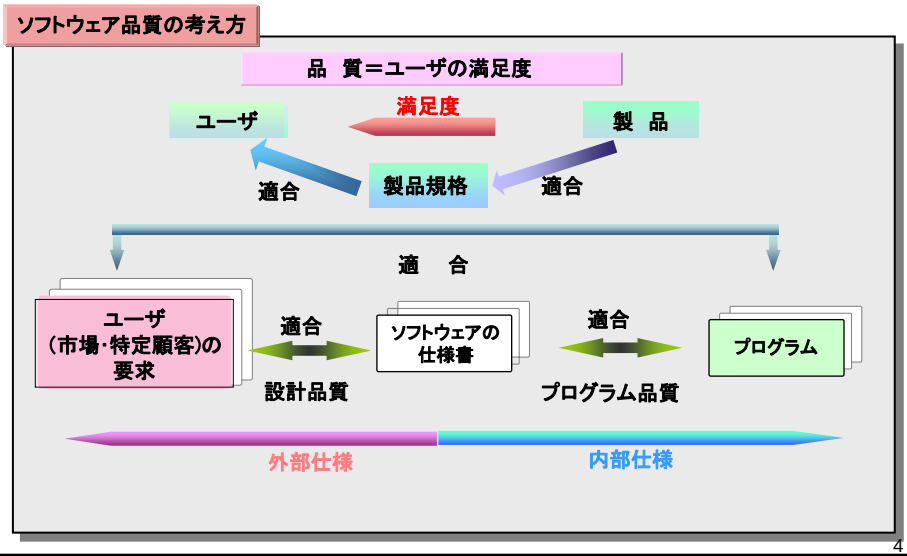
自己紹介

1962年(株)日立製作所に入社、
1965年以来 ソフトウェア開発事業部門、情報システム構築部門、
および関連会社((株)日立システムアンドサービス、(株)日立システムバリュー)において、ソフトウェアの品質保証、ソフトウェアテスト、SPI、PM普及などに従事。
2007年、NARAコンサルティングを立上げ、ソフトウェア開発のコンサルタント業務を展開、現在に至る。

コンテンツ

１．ソフトウェア品質二つの側面

ソフトウエアを中心とするシステム開発において品質を考える際は、狭義と広義の両面から考える必要が有る。狭義には、ソフトウエア仕様書に記述（定義）された機能が実現されている事の確認であり、広義にはユーザ要求への適合度、すなわちシステム完成度の確認である。



＜ソフトウェア検証の二つの条件＞

1、設計仕様がユーザ要求に合致していること。
2、プログラムが設計仕様を実現化し、仕様通りに正しく動作すること。

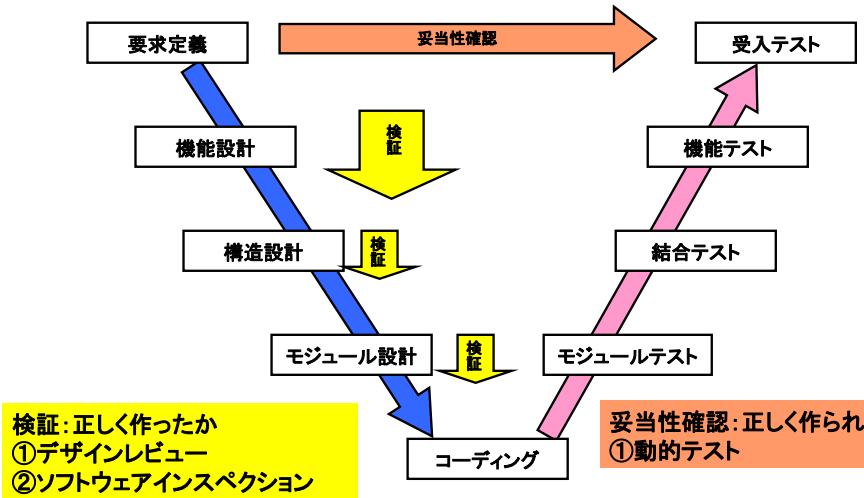
設計品質は外部仕様の解釈により定まり、
プログラム品質は内部仕様の正確性によって定まる。

最終的にはユーザ要求とプログラムが
適合していること

2. ソフトウェアの特徴

項　番	特徴と問題点	信頼性向上への配慮
1	＜論理の集合であること＞ ・論理の正確な設計が困難 ・論理の信頼性の高いテストが困難 ・正確な開発規模の見積りが困難	(1)構造設計とそれに基づくレビュー (2)システムマチックなテスト
2	＜目に見えないこと＞ ・品質管理が困難 ・工程管理が困難	(1)品質のビジュアル化 (2)開発工程のビジュアル化
3	＜個人への依存が高いこと＞ ・個人差が大きい ・多人数の共同作業	(1)開発手法の標準化と自動化 (2)再利用技術 (3)教育
4	＜ユーザニーズと直結していること＞ ・ユーザニーズの正確な理解が困難 ・使用条件の正確な把握が困難 ・なかなか仕様が決まらない ・システムは生き物である（成長する）	(1)要求仕様分析手法 (2)実使用条件でのテストによる検証 （ex. System Simulation Test）

３．　Ｖ＆Ｖ（検証と妥当性確認）



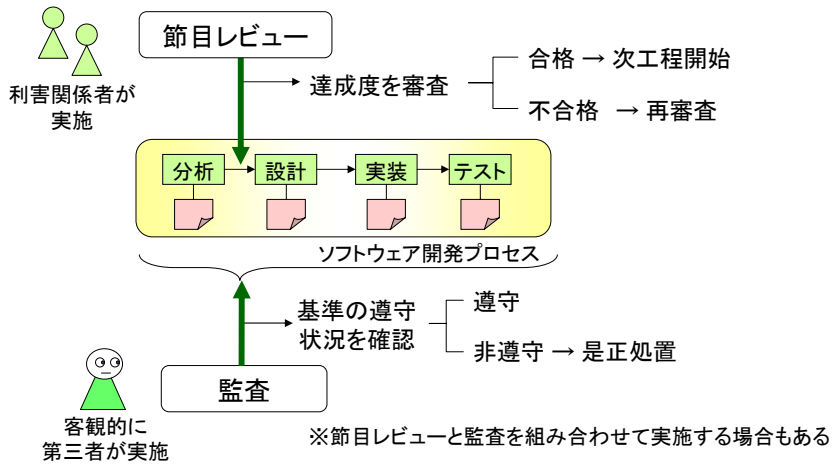
Ｖ＆Ｖ　（verification＆validation）

検証と妥当性確認

- 「検証」と「妥当性確認」とは？
  - 検証（Verification）
    - 「検証」の目的は、選択された作業成果物が、指定された要件を満たすことを確実にすることである
      - 「それを正しく構築した」ことを保証する　**making the thing right**
  - 妥当性確認（Validation）
    - 「妥当性確認」の目的は、成果物または成果物構成要素が、意図した環境に設置されたときにその使用意図を充足することを実証することである
      - 「正しいものを構築した」ことを保証する　**making the right thing**
- IV＆Vと呼ばれることも
  - 検証の独立性を重視する観点から、IV&V（Independent Verification and Validation：独立検証と妥当性確認）と呼ばれることもある

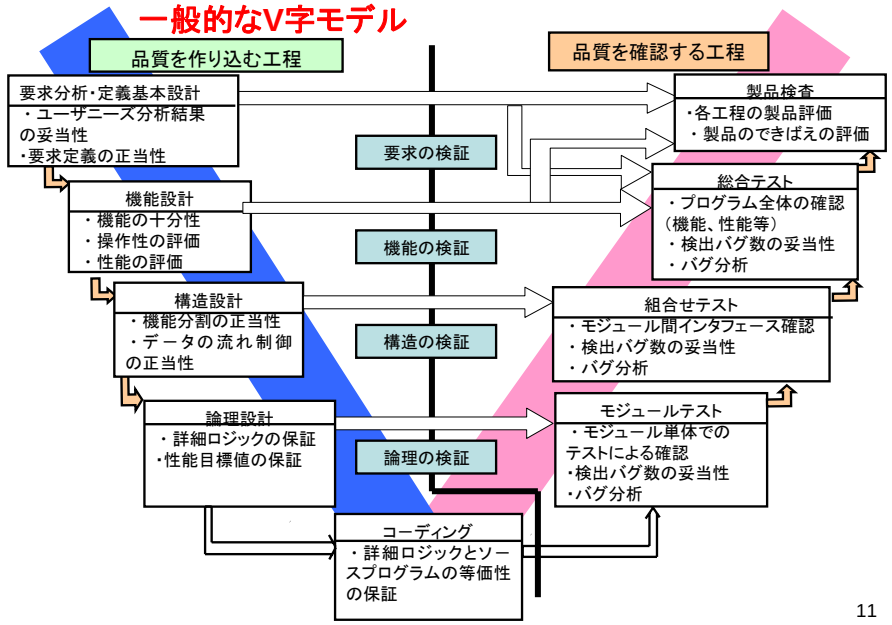
コンテンツ

４．検証手段としてのレビューの位置づけ



品質作り込みのプロセスが、
機能していることの確認が重要

5. V字モデル（妥当性確認）

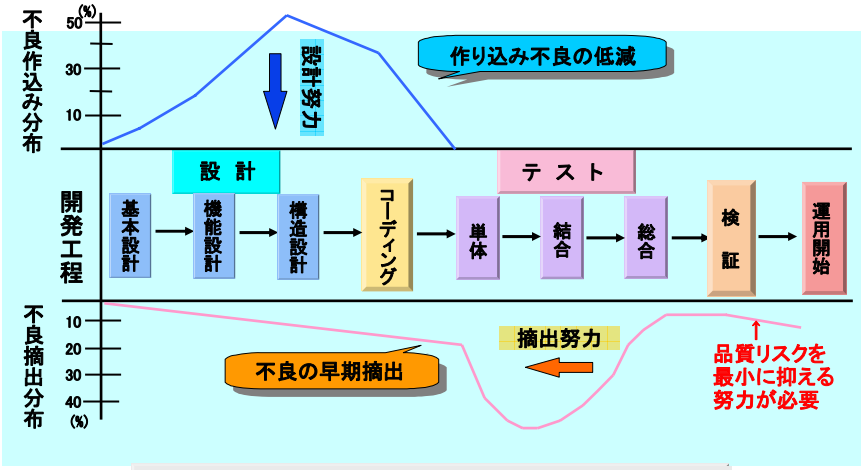


11

出展：保田勝通　奈良隆正著『ソフトウェア品質保証入門』

6. 検証手段としてのテストの必要性と定義

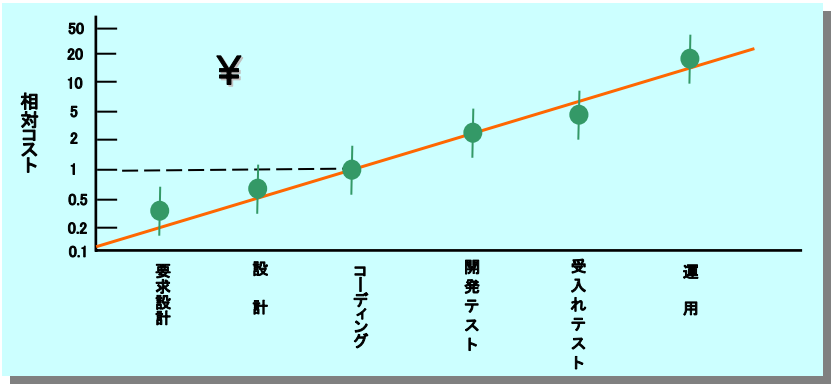
品質の作りこみと不良の摘出（テスト）



●設計段階のレビューを重視して作り込み不良を少なくすること。
（作り込んだ工程で不良を検出する。）
●作り込まれた不良は出来るだけ早い時期に摘出すること。

12

6．1　不良修正のコスト



コーディング工程で作り込んだ不良

テストで発見　：相対コストは、約2倍

運用テストで発見：相対コストは、約10～50倍

一番コストがかかるのは、下流のテスト工程での不良発覚
（社外事故の場合、コストはＭａｘとなる。）

13

出展：日本規格協会「ソフトウェア品質ガイドブック」

コンテンツ

14

7. レビューの技術

7．１　レビューの定義と概要

（1）デザインレビューとは

－設計工程の主要な切れ目で行われる見直し作業

－審査で設計上の欠陥を摘出、修正と改善余地の検討

（2）デザインレビューの目的

①成果物の品質評価（不良摘出を含む）による品質向上

②成果物の評価による進捗状況の把握

③上流工程での品質作り込みによる、手戻り削減と生産性向上

④レビュー結果を開発技術／技法やプロセス改善に役立てる

⑤担当者以外の仕様理解者を増やす、レビュー参加者のOJT

（3）デザインレビューの要点

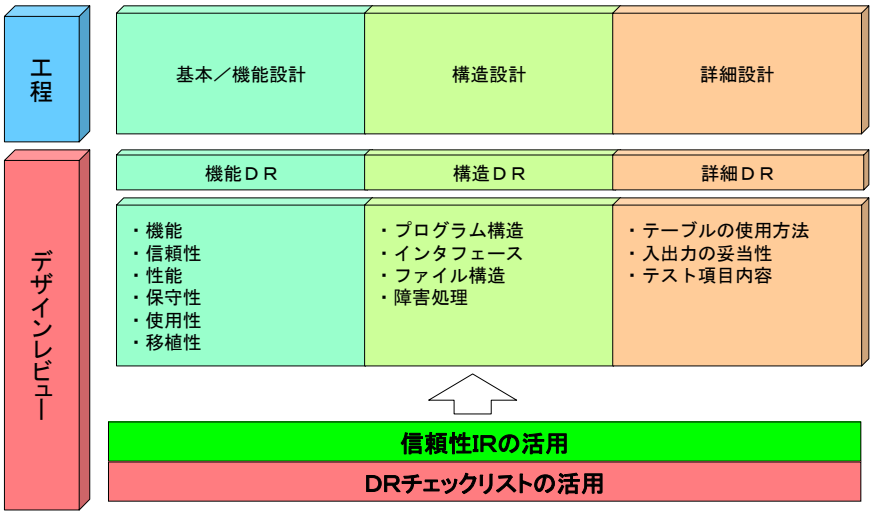


図7.1　デザインレビュー（ＤＲ）の要点

（4）デザインレビュー成功のポイント

①開発スケジュール表にレビュー工程と日程を明示する

②タイムリーに明確な仕様書（成果物）が作成されること

③適切なメンバーの参画

④レビューチェックリストの整備と活用

⑤過去の不良事例の整備（編集）、容易な検索、活用

⑥レビューの事前準備として各種資料が必要

7．2　レビューの実際

7．2．1　色々なレビュー

No.	名称	目的
1	ウォークスルー	成果物のエラーの発見
2	ピアレビュー	不明点や間違（勘違）いの発見
3	ソフトウェアインスペクション（Faganのインスペクション）	対象の合否判定とプロセス改善
4	公式レビュー（節目レビュー）	プロジェクトとしての合否判定
5	コードインスペクション	コード上のエラーの発見

7．3　ウォークスルー

（1）ウォークスルーとは⇒成果物のエラーの発見

- □ マシンテストに先立って同様の方法でレビューする、すなわち手によるシミュレーションを行うこと
- □ 何らかの成果物を開発チーム内の仲間のグループでレビューすること ⇒ 公式レビューとの違い
- □ 討議内容、制限時間、合意手続きなどの運営方法は公式レビューと同じように決めておく

（2）実施方法

- □ 説明者（＝開発者）、ファシリティータ、レビューア、記録者の5～6名で構成、担当者が説明、他の参加者がチェックする。実施時間は1～2時間が望ましい
- □ 原則、参加者に管理者は入れない ⇒結果を個人評価に利用しないための配慮
- □ レビューアは、エラーを発見できる能力を持つと判断されたひと

7．4　ピアレビュー

（1）ピアレビューとは⇒不明点や間違（勘違）いの発見

- □ チーム内の同僚間で行う不明点や間違い、バグなどの検出を目的とするレビューのこと、一般に作業成果物の評価は含まない
- □ レビューアは担当者以上の能力をもつものが作業経緯や結果精査する
- □ 誤りの検出だけでなく、さらにより良い成果物を生み出すための方法を提案し当事者間で議論する

（2）実施方法

- □ 開発の各プロセス（要求定義、設計、コード作成など）において、その途中段階や担当者の完了時点で必要に応じて適宜行う。
- □ 上司、管理者は参加しない、作業実績に対する評価は行わない

7．5　ソフトウェアインスペクション

（1）ソフトウェアインスペクションとは
⇒対象の合否判定とプロセス改善

- □ レビュー対象仕様書（成果物）と当該仕様書作成作業の入力資料（ソース文書）を比較しながらレビューを行う
- □ ソフトウェア要素が、その仕様を満足していること、適用標準に従っていることの確認、標準および仕様からの逸脱が無いこと
- □ ソフトウェア工学データの収集（例；工数、欠陥のデータ）

（2）実施方法

- □ 基本的な方法はウォークスルーと類似、但しルールは厳密
- □ モデレータと呼ばれる実施責任が、会議を計画／実施、レビューアの選定、結果報告書、指摘事項の修正確認まで全てを取り仕切る
- □ 結果の収集分析と有効活用（エラーの分類、タイプの分類、エラーの記録、をして開発者にフィードバック、類似エラーの摘出を支援）

7．6　公式レビュー（節目レビュー）

（1）公式レビューとは　⇒プロジェクトとしての合否判定

- □ 開発（プロセスの結果）の達成度合いを審査（合否判定）、当該工程のフェーズアウトと次工程の開始可否を判定する
- □ レビューの主体は開発部門であり、技術上の総合評価が主題であるが、審査は技術面のみならず、プロジェクトマネージメント面（管理面、進捗面など）も対象になる
- □ 参加者（レビューア）はプロジェクトの利害関係者（QA，PM，SE，営業、顧客　、など）全て、さらに外部の有識者や専門家

（2）実施方法

- □ 開発者（＝説明者）が説明、他の参加者がチェックし、不具合を指摘する
- □ 指摘事項の処置は、後日別途検討し、再審査を受ける
- □ 関連（必要）資料の事前準備と事前配布

　仕様書類のみならず検討に必要な資料（背景、機能概要、方式概要、など）

7．7　コードインスペクション

（1）コードインスペクションとは　⇒コード上のエラーの発見

- □ プログラムコードのレビューであり、机上デバッグの一方法論
- □ モデレータが居ることで個人のバラつきを排除し、レビューの質を向上する

（2）実施方法

- □ 説明者（＝開発者）はモデレータのもと、特定テスト項目、データを想定しプログラムの動作をコード上で追跡する
- □ 結果（バグ）の収集分析と有効活用（ソフトウェアインスペクションと同じ）により効果（効率）がさらに向上する

◆先人の教訓

（1）他のいかなる手法より欠陥除去の効果が大きい、除去率が常に60%を超えるのはこの手法だけである（Capers Jones）

（2）「コード審査」と題する論文では、これで見つけたエラーは全体の80%を占めた、さらにFagan流の審査では開発資源の15%を費やすがトータル開発コストを正味25～30%を低減する（アランデービス著、ソフトウェア開発201の鉄則）

＜参考＞　レビューを評価するメトリックス例

1．レビュー実施を評価する指標

（1）レビュー実施率；　レビュー工数／あるべきレビュー工数

　　－ レビュー漏れ、無駄、過剰のチェック

（2）レビュー寄与度；　発見バグ（仕様書）／摘出目標としたバグ

　　－ 仕様書レビューの貢献度（コーディング前でバグ摘出）

（3）内容指摘率；内容指摘件数／全指摘件数

　　－ レビュー内容の良さ、悪さ（レビューの妥当性、70%）

（4）発見バグ達成率；　発見バグ／全摘出目標としたバグ

　　－ バグ摘出による残存バグの予想

2．レビューで品質を評価する指標

　－ 頁当りのドキュメント不良、ドキュメント検査回数など、

　　詳細は、次の「設計プロセスのメトリックス例」を参照

＜参考＞　設計プロセスのメトリックス例

項番	工程	品質指標	単位	管理レベル	備考
1	基本設計	1. ドキュメント不良件数	件	プログラム単位	
		2. ドキュメント不良件数／頁	件／頁	プログラム単位	
		3. DR 指摘件数／頁	件／頁	プログラム単位	
		4. 投入工数／頁	人 H／頁	プログラム単位	
		5. 検完遅延日数	日	プログラム単位	
		6. ドキュメント検査実施回数	回	プログラム単位	
2	機能設計	1. ドキュメント不良件数	件	プログラム単位	
		2. ドキュメント不良件数／頁	件／頁	プログラム単位	
		3. DR 指摘件数／頁	件／頁	プログラム単位	
		4. 投入工数／頁	人 H／頁	プログラム単位	
		5. 検完遅延日数	日	プログラム単位	
		6. 変更指示件数	件	プログラム単位	
		7. ドキュメント検査実施回数	回	プログラム単位	
3	構造設計／詳細設計	1. ドキュメント不良件数	件	プログラム単位	
		2. ドキュメント不良件数／頁	件／頁	プログラム単位	
		3. DR 指摘件数／頁	件／頁	プログラム単位	
		4. 投入工数／頁	人 H／頁	プログラム単位	
		5. 検完遅延日数	日	プログラム単位	
		6. テスト項目数／KS	個／KS	ﾌﾟﾛｸﾞﾗﾑ、ﾓｼﾞｭｰﾙ単位	
		7. 変更指示件数	件	プログラム単位	
		8. ドキュメント検査実施	回	プログラム単位	

25

コンテンツ

26

8.　テストの概要

8．1　ソフトウェアテストの定義

（1） IEEEの定義

「（テスト(testing)とは）、手作業又は自動化された方法で、それが規定された要求事項を満たすかどうかを検証し、あるいは期待される結果と実際の結果との際を識別するために、システム又はシステム構成要素を実行し、又は評価する過程。」（IEEE標準『ソフトウェア工学用語集（ANSI／IEEE Std 729)』）

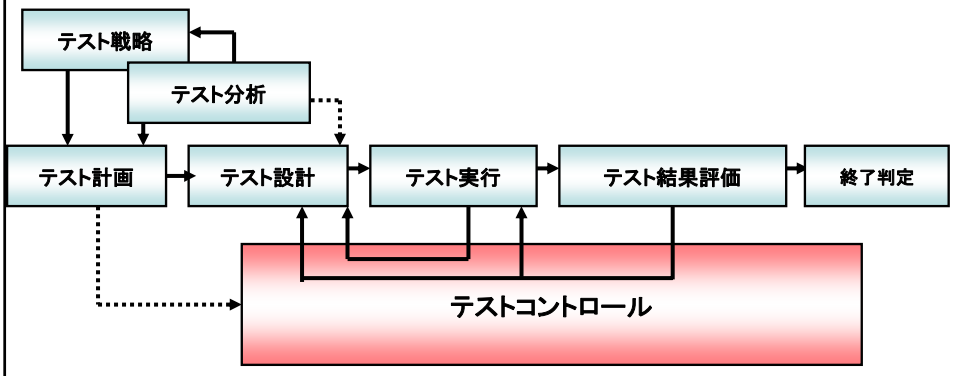
（2） G. J. Myersの定義

「テストとは、エラーを見つけるつもりでプログラムを実行する過程である。」（Myers, G.J.（著）『ソフトウェア・テストの技法』）

（3）玉井他の定義

「テストとは、エラーをなるべく見つけること、そしてエラーが見つからなければそのプログラムを品質に対する確信が増すこと、を目的として、プログラムを選んだデータを実行し、その結果を評価する作業である。」（玉井哲雄ほか『ソフトウェアのテスト技法』）

8．2　テストプロセス

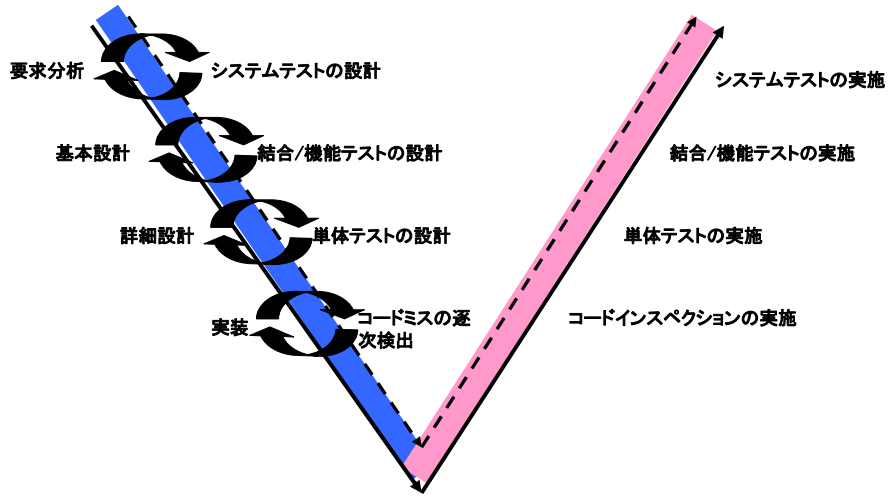


課題：テストアクティビティ早期開始

⇩

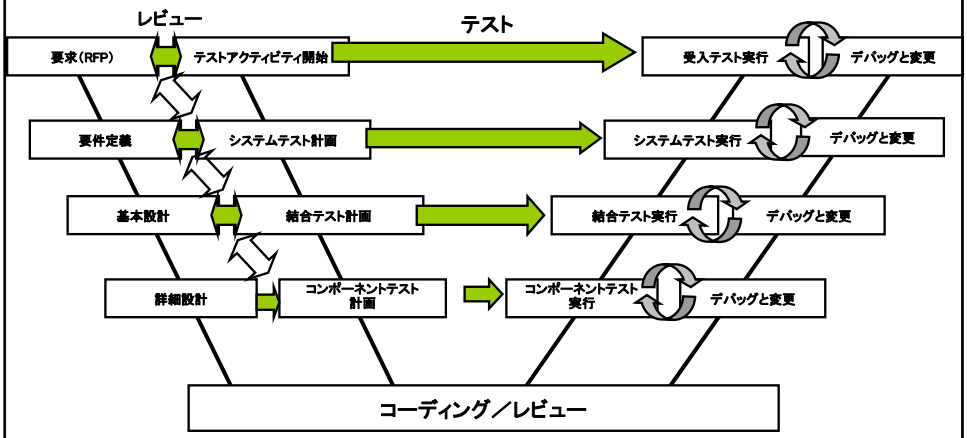
如何に担保するか！

8．3 ダブルV字モデル（テストファースト）



電通大　西先生　講演資料より

8．4 W-model（テストアクティビティの早期開始）

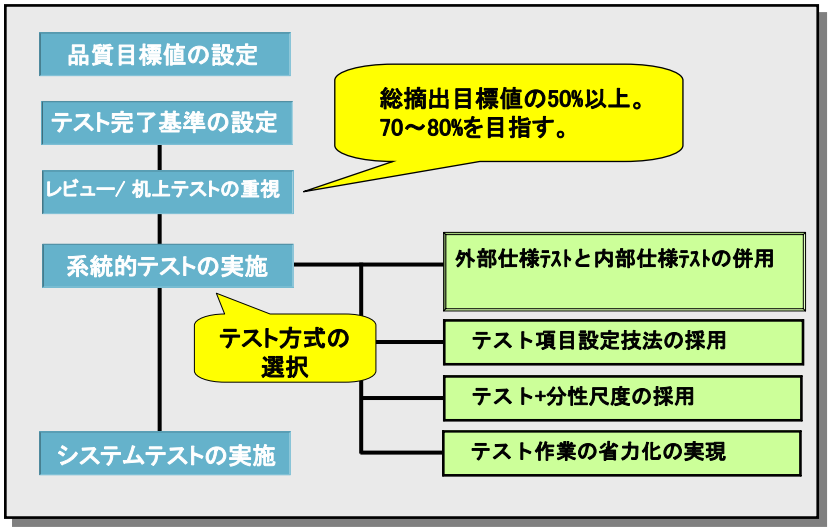


テストの関与を早め、テストフェーズの課題を取り除く

出展：JaSST’08チュートリアルTsuyoshiYUMOTO

補足1:テスト戦略策定の例

テスト分析が必要！



補足2:テスト分析とテスト設計の視点

（1）テスト分析の視点

①ソフトウェアの重要性（社会的影響、使われ方／局面、複雑さ）
②開発の難易度
③開発方法論
④開発期間

（2）テスト設計の視点

①ユーザの視点（どう使うか）
②バグの視点（バグで困ること、影響）
③仕様の視点（何を、何故）
④設計の視点（どのように実現したか）

補足３： 目標値設定（テスト工程）の具体例

ﾊﾞｸﾞ摘出／テスト項目設定目標値の例（KLOCベース）→80年代

単 体 テ ス ト	結 合 テ ス ト	総 合 テ ス ト
1.PCL件数： 100件／kloc 2.PCL質的内容 正常 ：70%以下 異常 ：10%以上 限界/境界：15%以上 ｲﾝﾀﾌｪｰｽ ：5%以上 3.バグ摘出件数： 8～10件/ kloc	1.CCL件数： 20～25件/ kloc 2.バグ摘出件数： 1～2件/ kloc	1.SCL件数： 5～10件/ kloc 2. バグ摘出件数： 0.5件/ kloc

＊ＰＣＬ：単体テスト項目、ＣＣＬ：結合テスト項目、 ＳＣＬ：総合テスト項目

補足４：目標値設定（テスト工程）の具体例

ﾊﾞｸﾞ摘出／テスト項目設定目標値の例（FPベース）→90年代

単 体 テ ス ト	結 合 テ ス ト	総 合 テ ス ト
1.PCL件数： 4.1件／FP 2.バグ摘出件数： 0.34件/FP	1.CCL件数： 1.0件／FP 2.バグ摘出件数： 0.04件/FP	1.SCL件数： 0.4件／FP 2. バグ摘出件数： 0.02件/FP

※この目標値は統計的手法により求めた値であり、各プロジェクトにおいては
その特性を考慮しプロジェクト毎に設定する必要がある。

＊ＰＣＬ：単体テスト項目、ＣＣＬ：結合テスト項目、 ＳＣＬ：総合テスト項目

補足5:テストの完了基準

（1） 基準1（摘出バグ）
バグ摘出目標値で設定したバグ数が摘出されていること。最低でも80%以上を目標とし、100%に達しない理由を評価しておくこと。

（2） 基準2（テスト項目）
テスト項目が目標値通りに設定され、全てのテスト項目が実施されていること。

（3） 基準3（バグ対策）
テストで摘出されたバグが、原則として全件修正されていること。修正を後工程に繰り越すときは、その理由と時期を明確にすること。

（4） 基準4（品質改善）
当該テスト工程で指摘されたバグを分析、評価し、それに基づく品質改善（追加テスト）が行われていること。この作業を後工程に繰り越すときは、その理由と時期を明確にすること。

補足6. テストの種類

テストの種別

1. 静的テストと動的テスト
2. ホワイトボックステストとブラックボックステスト
3. 網羅的テストとピンポイントテスト
4. フェーズドテスト：単体テスト、総合テスト、総合テスト

補足6．1　静的テストと動的テスト

静的テスト

プログラムの実行を伴わないで、プログラムのエラーや構造上の問題を検出する
（例：UNIXのLint、日立システムアンドサービスのInspect Pro等）

動的テスト

プログラムを実行させて、その正しさをチェックする。

通常テストと呼ばれるのは、この動的テストのことを言う。

補足6．2　ホワイトボックステストとブラックボックステスト

ホワイトボックステスト

プログラムの内部構造を調べて、論理が正しいか否かをチェックする。
（プログラムの内部構造を知ることができる場合のみ実施可能）

ブラックボックステスト

プログラムの内部構造には関知せずに、仕様通りの動きをするか否かをチェックする。

補足6．3　網羅的テストとピンポイントテスト

網羅的テスト

要求仕様や機能要求を全般的にわたり、網羅的にテストする。

ピンポイントテスト

経験などに基づきバグの存在しそうな部分を探し出し、そこを重点的にテストする。
（エラー推測法、探索的テスト　等）

補足6．4　フェーズドテスト

単体テスト

モジュール内のアルゴリズム、データ構造、実行パス、エラー処理パスのチェックを行う。

結合テスト

モジュール間のインタフェースのチェックを行う。

総合テスト

全てのモジュールを結合した最終的なシステムレベルでの実運用を想定した機能のチェックを行う。

コンテンツ

９．テスト技術（方法論）

（1） 2つのテストアプローチ

(a)外部仕様（ブラックボックス）テスト

－ ソフトウェアの一構成要素をブラックボックスと見なし、内部構造や論理にとらわれず外部仕様のみに従ってテスト項目を設定し、実施する

(b)内部使用（ホワイトボックス）テスト

－ ソフトウェアの内部構造や論理に着目しテスト項目を設定し実施する

(2)外部仕様テストと内部仕様テストの特徴

テスト項目の設計技法は、テストデータを何に基づいて作成するかという観点から、以下の２つに分けることができる。

ーブラックボックステスト（「外部仕様テスト」とも呼ばれる）ー

プログラムを１つのブラックボックス（すなわちプログラムの内部構造や論理を無視する）と見なし、テストデータを仕様書だけから引き出す方法である。

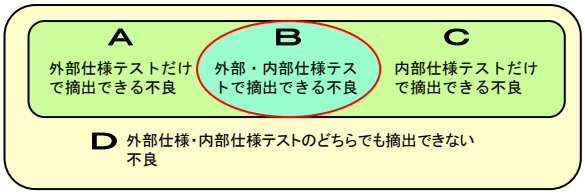
ーホワイトボックステスト（「内部仕様テスト」とも呼ばれる）ー

プログラムの内部構造や論理を調べることによってテストデータを引き出す方法である。

項番	テスト区分	基礎となる仕様	特徴	
			長所	短所
1	外部仕様テスト（ブラックボックステスト）	**ソフトウェアの外部仕様** ・プログラムまたはモジュールをブラックボックスとして見た仕様 ・ユーザからみた仕様	ユーザの立場からのテストができる	外部仕様に現れない内部仕様上の特殊な箇所のテストが漏れる
2	内部仕様テスト（ホワイトボックステスト）	**ソフトウェアの内部仕様** ・プログラムまたはモジュールの内部構造を規定する仕様 ・開発者からみた仕様	内部仕様上から網羅的にテストができる	外部仕様上規定されていながら実現されていない部分のテストが漏れる

43

(3)外部仕様テストと内部仕様テストの関係



A＋B＋C＋D：ソフトウェア不良の全体

A＋B　：外部仕様テストで摘出できる不良

B＋C　：内部仕様テストで摘出できる不良

A＋B＋C　：外部仕様・内部仕様テストで摘出できる不良

A　：外部仕様テストだけで摘出できる不良

B　：外部仕様・内部仕様テストで共通に摘出できる不良（内部仕様の欠落）

C　：内部仕様テストで摘出できる不良（外部仕様に対応しない内部仕様の不良）

D　：外部仕様・内部仕様テストでも摘出できない不良
（ユーザ要求の外部仕様での実現漏れ）

44

図9.1　外部仕様テストと内部仕様テストの関係

９．１　外部仕様（ブラックボックス）テスト

９．１．１　同値分割

（１）プログラムの有力域を有限数の同値のクラスに分割し、それぞれのクラスの代表的な値をテストする

（２）プログラムの入力条件を取り上げ有効な値の範囲（有効同値クラス）と、無効な値の範囲（無効同値クラス）に分割する事によりクラス分けを行う

９．１．２　限界値分析／境界値分析

（１）プログラムの入力領域、出力領域を同値クラスに分割

（２）それぞれの同値クラスの端がテスト対象となるような値を選ぶ

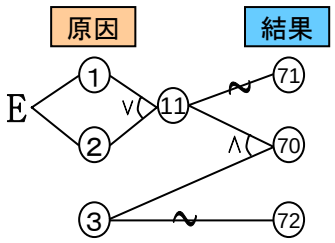
＜参考＞
同値分割、境界値分析の具体例を、付録１へ記載している。

９．１．３　原因－結果グラフ

（１）外部仕様から入力条件／環境条件（＝原因）と
　処理／出力（＝結果）の関係を論理値グラフで表す

（２）グラフからデシジョンテーブル（決定表）を作成し、テスト項目を設定する

（３）グラフ作成には4つの「制約条件」があり、この適用により最適な組合せが可能に成る



原因－結果グラフの例

テストNo. / 原因結果		1	2	3	4
1	第１列目の文字がＡ	1	0	0	
2	第１列目の文字がＢ	0	1	0	
3	第２列目の文字が 数字	1	1		0
70	ファイルが更新される	1	1		
71	Ｘ１２を印字			1	
72	Ｘ１３を印字				1

デシジョンテーブルの例

＜参考＞
上記(1)～(3)の具体例を、付録２へ記載している。

９．１．４　機能図式

機能図式法は，状態遷移図と入出力条件の組合わせで記述された決定表との組合わせで外部仕様を表現する技法である 。

（1）外部仕様から入力条件／環境条件（＝原因）と処理／出力（＝結果）関係を、「状態遷移図」とデシジョンテーブル（決定表）の組合せで表す⇒「これを機能図式」と呼ぶ

（2）機能図式からテスト項目を設定する

（3）機能図式作成には4つの「制約条件」があり、この適用により最適な組合せが可能になる

機能図式によるテスト項目作成例

「現金自動支払機」の仕様

① キャッシュカードが入力されると**「暗証番号を入力して下さい」**というメッセージが出力され，暗証番号入力待ちとなる。

② 暗証番号が入力されると，キャッシュカードに登録されている番号と一致しているかのチェックを行い，一致していれば，**「金額を入力して下さい」**と言うメッセージが表示され，金額入力待ちとなる。
なお，誤った番号が，3回入力されるとカードの登録取消しを行い**「処理を打ち切ります」**というメッセージを表示する。その後**「カードを入力して下さい」**のメッセージを表示しカード入力待ちとなる。

③ 金額が入力されると，残高とのチェックを行い入力金額が残高より小さいか等しい場合は，お金，支払い通知書及びキャッシュカードを戻し，**「カードを入力して下さい」**のメッセージを表示してカード入力待ちとなる。
入力金額が残高より大きい場合には，**「金額を入力して下さい」**というメッセージを表示して金額入力待ちとなる。

機能図式の例

＜状態遷移図＞

あるアクションをとるまで，一定の状態を保っているため，イベントの境目として考えやすいものを状態としてとらえる。今回の仕様として状態として考えやすいものに，次の３つがある。

Ｓ０：“カードを入力して下さい”を表示してカード入力を待つ。
Ｓ１：“暗証番号を入力して下さい”を表示して暗証番号入力を待つ。
Ｓ２：“金額を入力して下さい”を表示して金額入力を待つ。

これを状態遷移図で表わすと、下図のようになる。

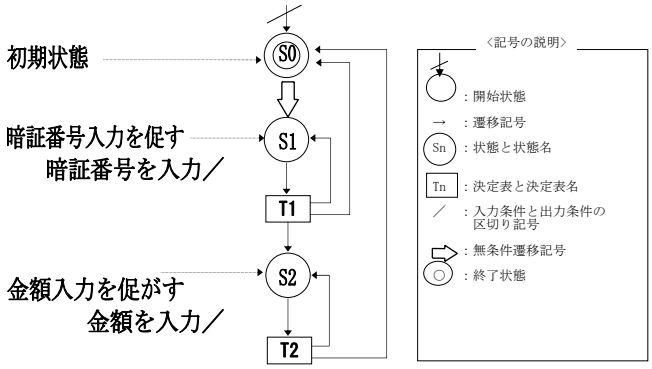


図9.1.4「現金自動支払機」の機能図式

T1決定表（S1の状態から始まる）

条　　件		テストケース					
入力条件	暗証番号は正しい	Y					
	暗証番号は誤り（１～２回目）		Y				
	暗証番号は誤り（３回目）			Y			
出力条件	“暗証番号を入力して下さい”を表示		*				
	“処理を打ち切ります”を表示しカードを返す			*			
遷移先	Ｓ０			*			
	Ｓ１		*				
	Ｓ２	*					

T2決定表（S2の状態から始まる）

条　　件		テストケース					
入力条件	金額≦残高	Y					
	金額＞残高		Y				
出力条件	“金額を入力して下さい”を表示		*				
	お金・通知書・カードを出力する	*					
遷移先	Ｓ０	*					
	Ｓ２		*				

9．1．5　要因分析法（直交配列表）

（1）プログラム動作時の入力条件（因子）とその状態（水準）を明確にし、それらの組合せでテスト項目を設定する技法

（2）組合せの効果的な絞込みに実験計画法の直交配列表を応用し効果的（最適）なテスト項目を設定する

直交表の例

L4直交表

L4直交表	A（因子）	B（因子）	C（因子）
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

C列の0・1：水準

因子・水準の対応表

因子	水準	直交表
太字	OFF	0
	ON	1
斜体	OFF	0
	ON	1
下線	OFF	0
	ON	1

文字修飾の因子・水準と直交表の中の数字との対応

L4直交表（割り付け後）

L4直交表	太字	斜字	下線	結果
テストNo.1	OFF	OFF	OFF	Test結果
テストNo.2	OFF	ON	ON	*Test結果*
テストNo.3	ON	OFF	ON	**Test結果**
テストNo.4	ON	ON	OFF	***Test結果***

ON・OFF：水準

出典）ソフトウェアテストPRESS Vol.2

9．1．6　エラー推測技法

（1）経験と直感から起こりそうなエラーを推測して、テスト項目を設定する方法、有りうるエラーを生みやすい状況を列挙してこれに基づいてテスト項目を抽出する

（2）経験を駆使し、エラーが起きた場合にどんな欠陥が被試験のコンポーネントやシステムの中に存在するかを予想して、その欠陥を検出するテストケースを設計する（出展：ISTQB-FL用語集）

9．1．7　探索的テスト

（1）発見されたバグ要因、原因、現象などで分析し、その結果を水平展開して同種のバグを摘出するテスト項目を設定する

（2）非公式テスト技法の一つ。テストを実施する過程で、テスト実施情報を活用しながらテスト設計を制御し、積極的に質の高い新しいテストケースを設計する（出展：ISTQB-FL用語集）

９．２　内部仕様（ホワイトボックス）テスト

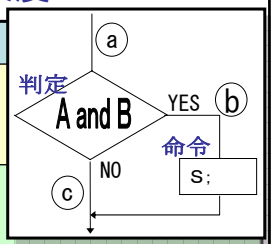
（１）基本的にはソースコードで実現された論理をベースとして、設定したある基準を満たすようなテストであり、論理網羅テストとも呼ばれる

（２）制御フローに基づく論理網羅の基準となるのが、テスト十分性評価尺度である

（３）さらに論理の流れを一つのパスとして捉え、それを網羅する制御パステストがある

９．２．１　制御フローに基づく十分性評価尺度

項番	評価尺度	内容
1	命令網羅	全ての命令が少なくとも1回は実行されるようにテストデータを作成する。
2	判定条件網羅（分岐網羅）	それぞれの判定条件が「真」と「偽」の結果を少なくとも1度ずつ持つようにテストデータを作成する。
3	条件網羅	判定におけるあらゆる条件で，すべての可能な結果を少なくとも1回は通るようにテストデータを作成する。
4	判定条件／条件網羅	1つの判定条件でのそれぞれの個別条件が，すべての可能な結果を少なくとも1回は通るようにし，かつ，それぞれの判定条件が可能な結果を少なくとも1回は通るようにテストデータを作成する。
5	複数条件網羅	それぞれの判定における個別条件の結果のすべての可能な組み合わせをテストデータとして作成する。

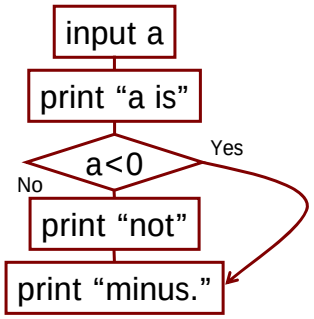


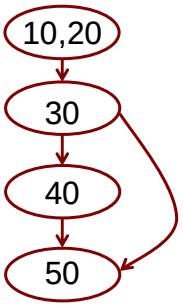
＜参考＞
「制御フローに基づく十分性評価尺度」の詳細を、付録3へ掲載している。

９．２．２　論理（ロジック）を網羅する制御パステスト

（１）　ロジックを網羅して全てきちんとモレなくテスト
- フローグラフを描いてロジックを抜き出す
  - フローグラフ：ノードとリンクで書かれた図
  - フローチャートや状態遷移図はフローグラフ

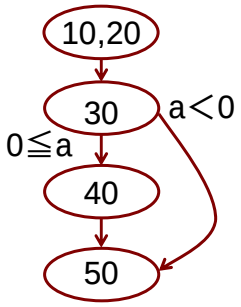
```
10  input a
20  print “a is”
30  if (a<0) goto 50
40  print “not”
50  print “minus.”
```





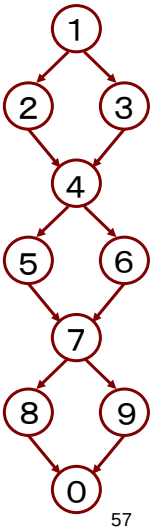
（２）制御パステストの設計：パスの網羅

- パスとは？
  - フローグラフ上の経路
  - プログラムのロジック
  - パスの数がテストの数に比例する
- パスを全て網羅するように
  テストを設計する
  - まずパスの一覧表を作る
    - 10→20→30→40→50
    - 10→20→30→50
  - 次にパスを通るデータを実装する
    - 10→20→30→40→50　　: a=0
    - 10→20→30→50　: a=-1



（3） どこまでテストすればいいのか？

- 全てのパスを組み合わせようとすると膨大になる
  - if文の数の累乗で増えていく
  - if文に含まれるandやorの数の累乗でも増えていく
    - 右のグラフでは8本のパスが必要となる
- 最低でもif文の両側の分岐は1度テストしよう
  - 分岐網羅（C1基準）
- 組み合わせが増えないように気を付けて開発する必要がある
  - KISS: Keep It Simple, Stupid!



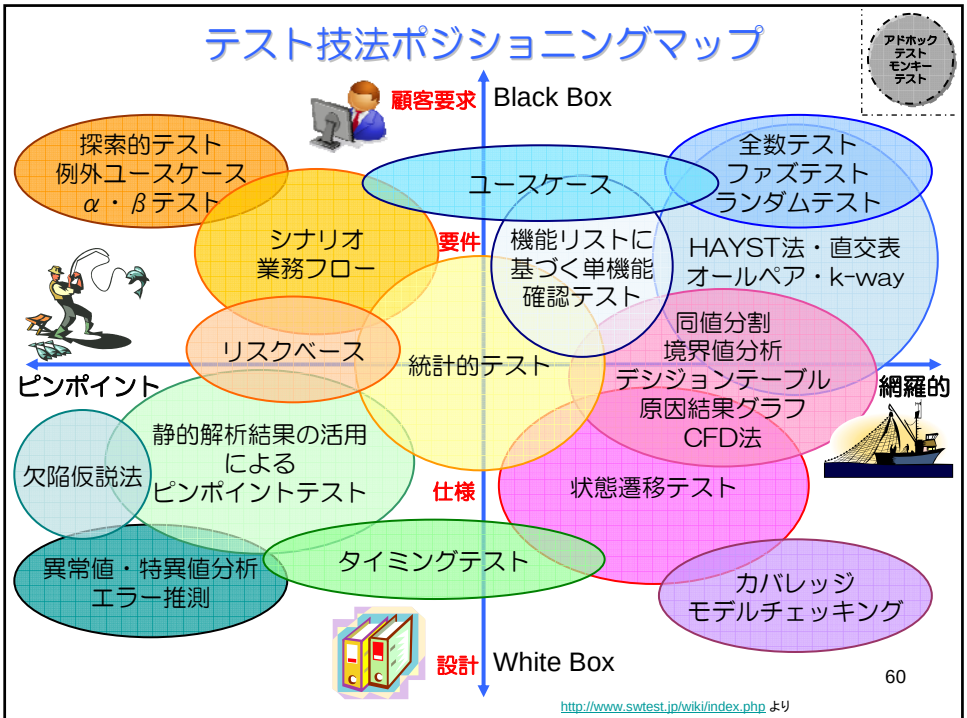
＜参考＞テストプロセス メトリックスの例

項番	工程	品質指標	単位	管理レベル	備考
1	単体テスト	1. 不良摘出件数（累積） 2. 不良摘出件数／KS 3. テスト項目進捗度 4. 不良摘出件数／投入工数 5. 不良摘出件数／マシン時間 6. テストプログラム消化率 7. 不良率（不良摘出件数／テスト項目数）	件 件／KS % 件／人H % % %	ﾌﾟﾛｸﾞﾗﾑ、ﾓｼﾞｭｰﾙ単位 モジュール単位 モジュール単位 モジュール単位 モジュール単位 モジュール単位 モジュール単位	成長曲線
2	プログラムテスト	1. 不良摘出件数（累積） 2. 不良摘出件数／KS 3. テスト項目進捗度 4. 不良摘出件数／投入工数 5. 不良摘出件数／マシン時間 6. テストプログラム消化率 7. 設計総合耐久テストの故障率 8. 探針での不良率 9. 探針での故障率 10. 探針での推定残不良数 11. 不良率（不良摘出件数／テスト項目数）	件 件／KS % 件／人H % % 件／H % 件／H 件 %	ﾌﾟﾛｸﾞﾗﾑ、ﾓｼﾞｭｰﾙ単位 モジュール単位 モジュール単位 モジュール単位 モジュール単位 モジュール単位 ｼｽﾃﾑ、ﾌﾟﾛｸﾞﾗﾑ単位 プログラム単位 プログラム単位 プログラム単位	成長曲線 （デバッグ効率）
3	製品検査	1. 不合格件数 2. 不合格件数／KS 3. 製品検査実施回数 4. 故障率（1/MTBF） 5. MTBD 6. 不良率 7. 不良摘出件数（累積）	件 件／KS 回 件／H H／件 % 件	ｼｽﾃﾑ、ﾌﾟﾛｸﾞﾗﾑ単位 プログラム単位 プログラム単位 プログラム単位 プログラム単位 プログラム単位 プログラム単位	（設計発見不良）

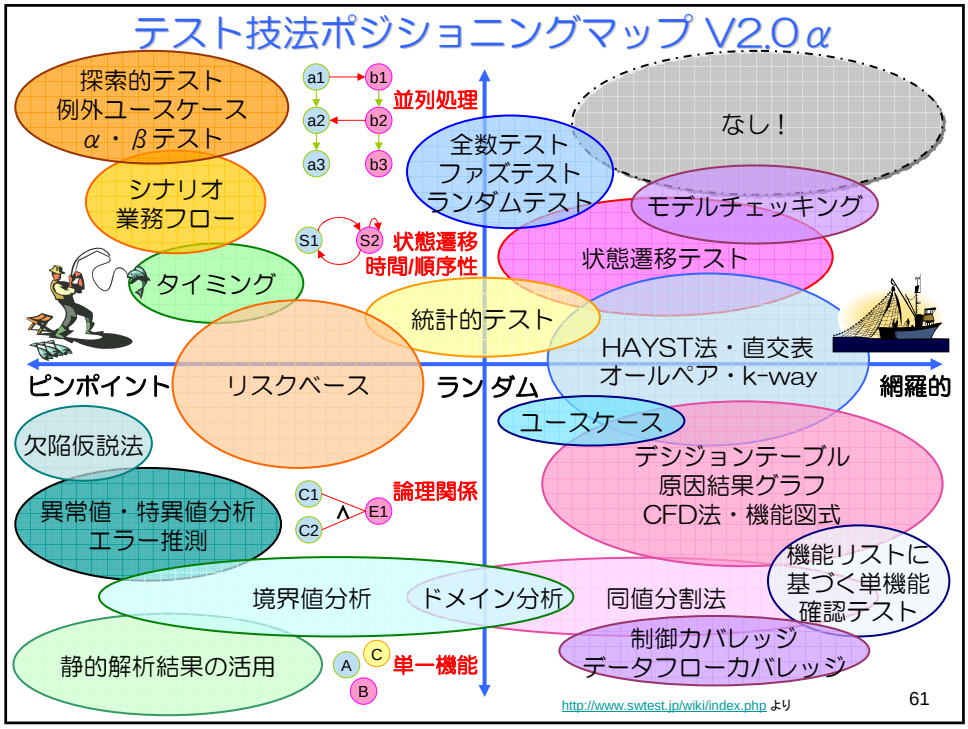
＜参考＞テストプロセスでの信頼性評価尺度

項番	品質特性	適用工程			計測尺度	定義
		開発	検査	稼動		
1	成熟性	○	○	○	障害密度	障害件数／規模(KS)
2		○	○		障害率	障害件数／テスト項目数
3		○	○	○	故障率	故障件数／マシン時間
4		○	○	○	平均故障発生間隔	稼働時間／故障件数
5		○	○	○	平均ダウン発生間隔	マシン時間／ダウン件数
6		○	○		障害収束率	累積摘出障害数／推定障害総数
7			○		探針不良率	探針不良件数／探針実施項目数
8		○	○		現象別障害比率	現象別障害件数／全摘出障害件数
9		○	○		原因別障害比率	原因別障害件数／全摘出障害件数
10	テスト品質	○	○		テスト密度	テスト項目数／規模(KS)
11		○	○		テスト実施率	実施済みテスト項目数／実施予定テスト項目数
12		○			テストカバレージ率(C0)	実施済みステートメント数／全ステートメント数
13		○			テストカバレージ率(C1)	実施済み分岐数／全分岐数
14	可用性		○	○	稼動率	実稼動総時間／予定稼動時間
15			○	○	平均復旧時間	復旧に要した総時間／ダウン回数
16				○	重症不良件数頻度	件／月

テスト技法ポジショニングマップ



http://www.swtest.jp/wiki/index.php より



http://www.swtest.jp/wiki/index.php より

テストのトピックス

1. ISTQB　Foundation Level シラバス、用語集
   （技術標準：ISTQB）
2. マインドマップから始めるソフトウェアテスト（書籍：技術評論社）
3. ソフトウェアテスト　HAYST法（書籍：日科技連）
4. オールペア法とPICT（技法：ボランテア（WEB））
5. TestLink（ツール：TEF）
6. TEF　（Testing Engineer's Forum）（コミュニティ）
7. 現場で使えるソフトウェアテスト（Java編）（書籍：翔泳社）
8. ソフトウェアテスト入門（書籍：技術評論者）
9. ソフトウェアテストの技法　第2版（書籍：近代科学社）
10. ソフトウェアテストの基礎（ISTQB-シラバス準拠）（書籍：BNN新社）

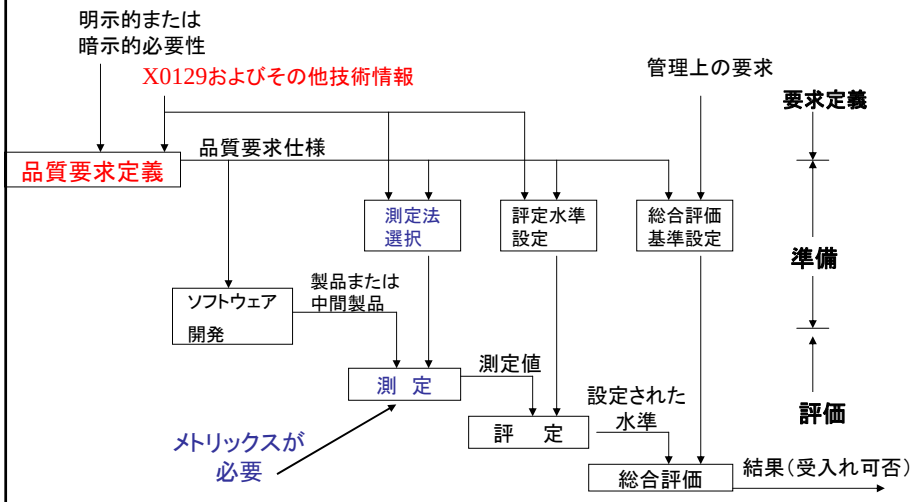
コンテンツ

10. 品質評価の技術

「測れないものは制御できない」(トム・デマルコ、Controlling Software Projects; Englewood Cliffs)のであり、計測はプロジェクト管理、品質管理のみならず、あらゆる管理(制御)の基本である。

ここでは、階層構造をもった品質特性を主体とする品質モデルと、品質評価プロセスをモデル化した品質メトリクスのメタモデル紹介する。

ISO/IEC9126　品質評価プロセスモデル



出典）東基衛（他編）:『ソフトウェア品質評価ガイドブック』、日本規格協会、1994年。

ソフトウェア品質特性（ISO/IEC 9126シリーズ）①

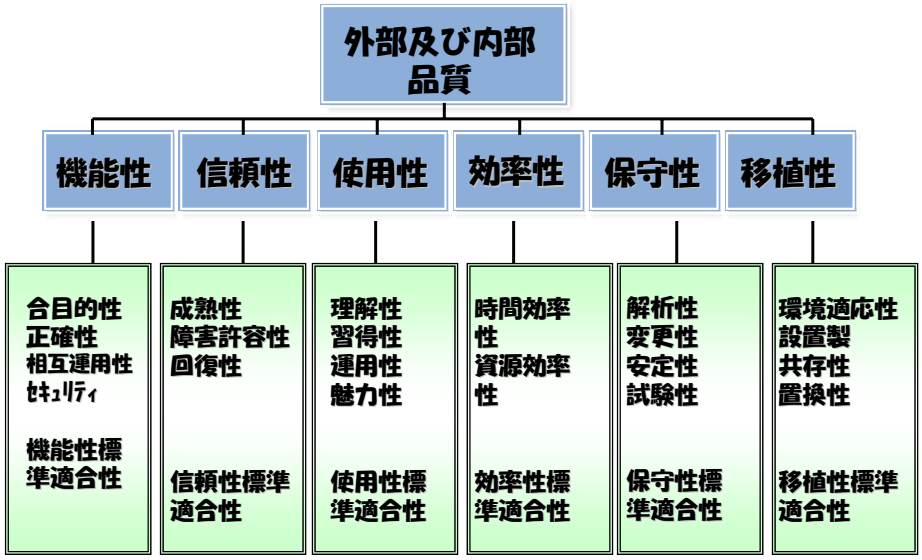


図10.1　外部及び内部品質のための品質モデル
［出典：JIS X 0129-1:2003 ソフトウェア製品の品質 － 第1部：品質モデル，p. 9　図4］

【参考】ソフトウェア品質特性（ISO/IEC 9126シリーズ）②

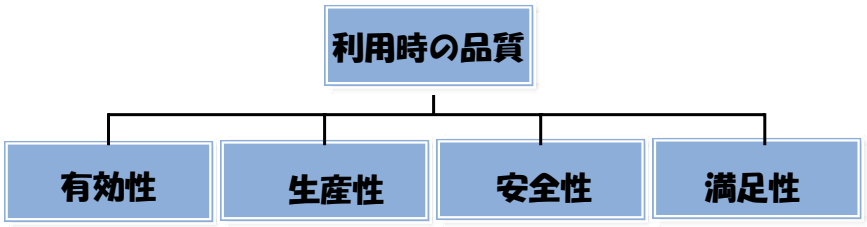


図10.2　利用時の品質のための品質モデル

［出典：JIS X 0129-1:2003 ソフトウェア製品の品質
－ 第1部：品質モデル，p. 13図5］

メトリックスとは

- “もの”の測定可能な特徴を属性と呼び、属性を測定する方法と尺度を合わせた概念の集合
- 2種類のメトリックス
  - プロダクトメトリックス；製品の属性を測定する
    - ソフトウェア品質メトリックス、規模メトリックスなど
  - プロセスメトリックス；プロセスの属性を測定する
    - 開発プロセスの個々の作業や手順全体に関するメトリックス
    - 作業に影響を与える人や組織といった開発基盤に関するメトリックス
- メトリックスを用いる目的；
  - 製品やプロセスの品質を定量的に把握．評価し、継続的に改善すること
  - 製品とプロセスの両方に焦点を当てて、改善することが重要
  - どの様なメトリックスを用いるかは、何を目的にどの様な目標を設定するかで決定

（ 出展：SQuBOKガイド）

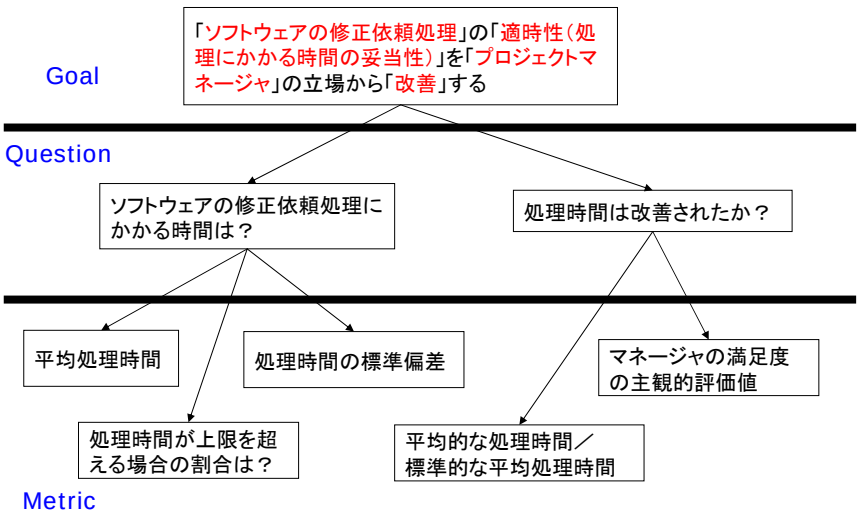
＜参考＞
ISO/IEC9126対応のメトリックス例を、付録４へ掲載している。

【参考】 メトリックス定義の代表的な技法

- GQMパラダイム
  - “計測はトップダウンに行われるべきである”
    - まず計測目標があって，その目標を遂行するために尺度（メトリクス）が定義され，計測されるべき
  - Basili教授らによって提案された総合的なソフトウェア計測の枠組み
  - “データ分析は何らかの目的や仮説に基づいて行われるべきである”
    - 例）コスト予測，コスト改善，品質改善　etc.

＊井上克郎，松本健一，飯田元　著「ソフトウェアプロセス」 共立出版，2000.

GQMモデル例＊



出展：EASEプロジェクト

ソフトウェア測定の考え方（出展：ＳＱｕＢＯＫガイド）

- **測定にあたり；**
  - 目的、方法（操作の場合）、尺度、活用のプロセスを明らかにする
  - 測定目的とメトリックスを結びつけることが重要（例：GQM）
  - 属人性を排した形で測定方法を定義することが重要
  - 名義尺度、順序尺度、間隔尺度、比率尺度のどれを用いるか定義

- **測定プロセスの共通的な枠組み（ISO/IEC 15939:2002）；**
  - 測定に対するコミットメントの確立および保持
  - 測定プロセスの計画
  - 測定プロセスの遂行
  - 測定の評価

● **測定プロセスとは、測定の目的や方法、尺度の定義、選択、適用を改善するために必要な手順**

測定理論　～用語の定義(1)（ISO/IEC:15939）

- **測定尺度；**
  - 測定可能な特徴を示す属性を計量するもの
    - 一定の尺度で物事を測定することにより、客観的な評価や判断を可能とる
- **品質メトリックス；**
  - ソフトウェアの品質特性、副特性を定量的に計測するもの
    - ソフトウェア製品の品質特性、副特性を定義し、測定、評価を可能とする
- **測定値；**
  - 属性に割り当てられた「値」（直接測定値と間接測定値）
    - 測定可能な特徴を値で示し、測定対象の実体や状況を具体的に把握する
- **指標；**
  - 予測や見積り、評価等の情報ニーズに基づいて示す数値または変数
    - 予測や見積りの際の目安や補助、組織やプロジェクトの達成度合いの評価基準に用いる
- **評定水準；**
  - 測定尺度を分類するために使われる順序尺度上の点
    - ソフトウェア種別や利用者の要求等を考慮して、評価目的ごとの品質要求水準を明確にする

測定理論　～用語の定義(2)（(ISO/IEC:15939)

- **評価基準;**
  - プロセス、製品、プロジェクト、または資源の価値や品質を評価するときの基準
    - 品質特性または品質副特性の評定水準に基づいて、ソフトウェア製品品質の総合評価が客観的に行えるようにする
- **測定プロセス;**
  - メトリックスを実際に計測するプロセス
    - 開発済みのソフトウェアメトリックスを計測し、当該ソフトウェア自体の評価、他ソフトウェアとの比較、今後の保守のスケジュール、投入人月の見積りに使用する。
    - 開発中のソフトウェアメトリックスを計測し、開発プロセスを動的に制御しながら、ソフトウェアを計画どおりに効率よく開発する「ソフトウェア開発の計器飛行」を可能にする
- **評価プロセス;**
  - ソフトウェアメトリックスを使用して計測した結果を評価する
    - ソフトウェア製品の品質評価を計画的に実施して、客観的、効率的な評価が行えるようにする

ソフトウェア測定まとめ

①計測は品質管理、プロジェクト管理のみならず、あらゆる管理(制御)の基本である。

②計測を確実にするためには、計測が容易であること、計測実績データの蓄積、評価が重要である。

③測定にあたり、目的、方法、尺度、活用のプロセスを明らかにすることが重要である。

計測事象＝因果関係＋バイアス＋偶発性

（出展;長崎大中村先生ーJASPIC講演より）

ソフトウェアエンジニアは
常にテストを意識し、テストを勉強
することで、優秀なソフトウェア開発者
になれる。

ご清聴ありがとうございました。

付録１

同値分割、境界値分析の例題

－int型の引数aがあります
－モジュール内にはif（a<0）と if（7<a）という
２つの条件文があります
・まず同値クラスを挙げてみましょう。
・次に同値クラスの境界値を挙げてみましょう

【説明】
解答は以下のとおり
同値クラスは以下の３つ

INT_MIN≦a＜0	0≦a≦7	7＜a≦INT_MAX

もともとは以下の4つを組み合わせて3つになった

INT_MIN≦a＜0	0≦a≦INT_MAX
INT_MIN≦a≦7	7＜a≦INT_MAX

境界値は | INT_MIN | -1 | 0 | 7 | 8 | INT_MAX | の6つ

演算子のバグ（＜を≦など）、範囲のバグ（a＜１など）、
型指定のバグ（intをより小さい型にするなど）、
int型の範囲の認識のバグ（処理系を16bitでなく32bitと思っていたなど）

（例題の出展：sessame初級テキスト）

付録２

原因－結果グラフの例

◆仕様◆

第1列目の文字は”A”か“B”でなければならない。

第2列目の文字は数字でなければならない。

この状況においてファイルの更新が行われる。

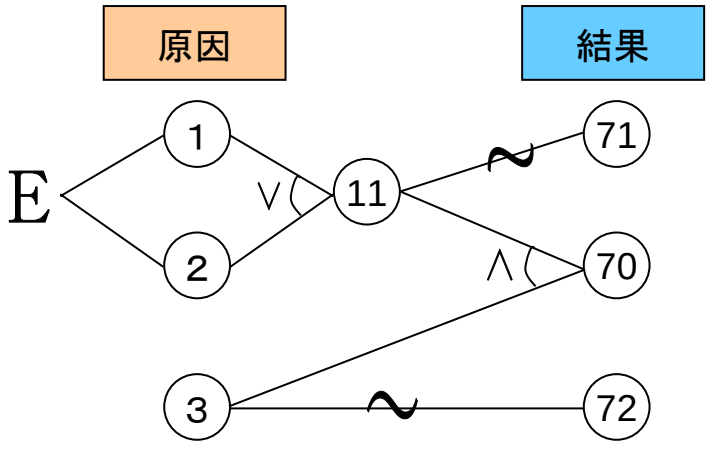
最初の文字が正しくなければX12のメッセージを印字する。

2番目の文字が数字でなければX13のメッセージを印字する。

原因は	結果は
１－第１列目の文字が“A”	70－ファイルが更新される
２－第１列目の文字が“B”	71－メッセージX12が印字される
３－第２列目の文字が数字	72－メッセージX13が印字される

付録２

原因－結果グラフ



付録２

デシジョンテーブル作成手順

１．１つの結果を選ぶ
２．グラフをこの結果から逆にたどり、この結果が存在状態（１で表す）になる原因の組合せを全てみつける
３．原因の組合せを表現するためのデシジョンテーブルの例をつくる
４．他の結果についても同様な操作を行い、デシジョンテーブルを完成させる

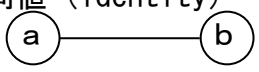
デシジョンテーブル

	テストNo. / 原因結果	1	2	3	4
1	第１列目の文字がＡ	1	0	0	
2	第１列目の文字がＢ	0	1	0	
3	第２列目の文字が 数字	1	1		0
70	ファイルが更新される	1	1		
71	Ｘ１２を印字			1	
72	Ｘ１３を印字				1

付録２

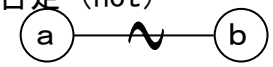
原因－結果グラフの記号

①同値（identity）



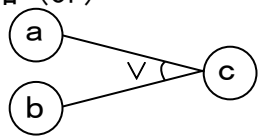
a=1 → b=1
a=0 → b=0

②否定（not）



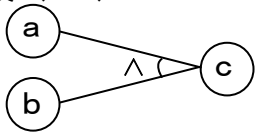
a=1 → b=0
a=0 → b=1

③和（or）



a=1
または → C=1
b=1
a=b=0 → c=0

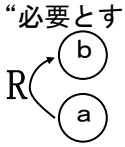
④積（and）



a=b=1 → c=1
a=0
または → C=0
b=0

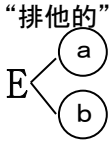
付録２
制約条件

① “必要とする”



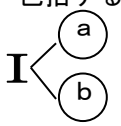
ａ＝１ならば
ｂ＝０となることは
ありえない

② “排他的”



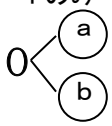
ａ＝１ならばｂ＝０
ｂ＝１ならばａ＝０

③ “包括する”



ａ＝１またはｂ＝０
（ａ＝ｂ＝０はありえない）

④ “１のみ”

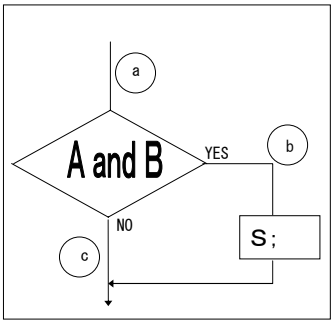


ａ＝１かつｂ＝０
または
ａ＝０かつｂ＝１、
（ａ＝ｂ＝０、 ａ＝ｂ＝１はありえない）

付録３
制御フローに基づく十分性評価尺度（1）

＜命令網羅＞

全ての命令が少なくとも１回は実行されるようにテストデータを作成する。



命令を網羅するには，ⓐ－ⓑのルートを通過するようにテストデータを作成すれば良い。

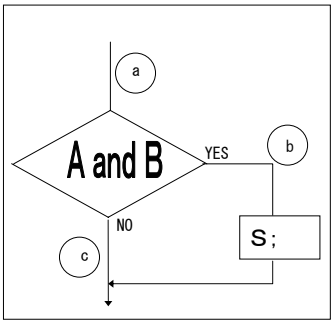
この基準からはⓐ－ⓑのルートを通過するテストデータは作成されないなど，一般的には網羅基準としては弱い。

（注）A，B：条件式
S　：命令

付録３
制御フローに基づく十分性評価尺度（２）

＜判定条件網羅（分岐網羅）＞

それぞれの判定条件が「真」と「偽」の結果を少なくとも１度ずつ持つようにテストデータを作成する。



判定条件を網羅するためには，判定条件の結果「真」の時と，「偽」の時を確かめるテストデータを作成すれば良い。すなわち，
「Ａ ａｎｄ Ｂ」が「真」→Ａ：「真」，Ｂ：「真」

「Ａ ａｎｄ Ｂ」が「偽」→Ａ：「真」，Ｂ：「偽」
Ａ：「偽」，Ｂ：「真」
Ａ：「偽」，Ｂ：「偽」

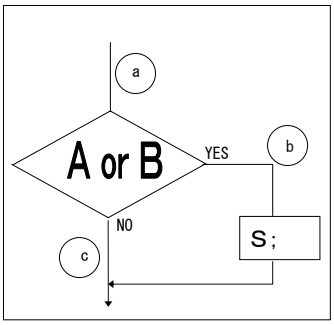
という２つのテストデータを作成し，ⓐ－ⓑとⓐ－ⓒのルートを通過させるようにすれば良い。

この基準は，個々の条件の全ての結果を確かめるものではない。

付録３
制御フローに基づく十分性評価尺度（３）

＜条件網羅＞

判定におけるあらゆる条件で，すべての可能な結果を少なくとも１回は通るようにテストデータを作成する。



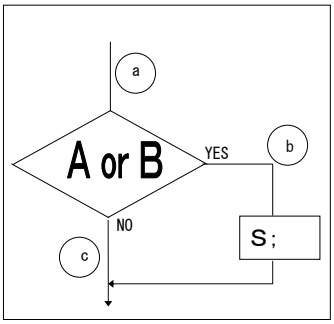
判定条件を構成する個々の条件（左図のＡとＢ）に着目し，それぞれの条件が「真」または「偽」となるようにテストデータを作成すれば良い。

例えば，
Ａ：「真」，Ｂ：「偽」
Ａ：「偽」，Ｂ：「真」
という２つのテストデータを作成すれば，個々の条件は網羅される。

付録３
制御フローに基づく十分性評価尺度（４）

＜判定条件／条件網羅＞

１つの判定条件でのそれぞれの個別条件が，すべての可能な結果を少なくとも１回は通るようにし，かつ，それぞれの判定条件が可能な結果を少なくとも１回は通るようにテストデータを作成する。



前述の、条件網羅のテストデータは
ⓐ－ⓒ のルートを通過していない。

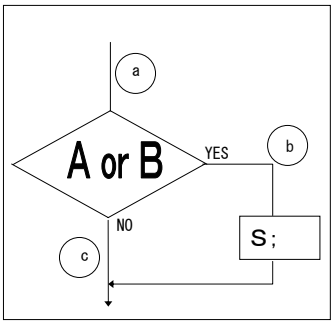
判定／条件網羅では，判定条件の結果が「真」の時と「偽」の時を確かめ，かつ，個々の条件が「真」または「偽」となるようにテストデータを作成する。

すなわち
Ａ：「真」，Ｂ：「真」
Ａ：「偽」，Ｂ：「偽」
という２つのテストデータを作成すれば，判定条件が網羅でき，かつ個々の条件も網羅できる。

付録３
制御フローに基づく十分性評価尺度（５）

＜複数条件網羅＞

それぞれの判定における個別条件の結果のすべての可能な組み合わせをテストデータとして作成する。



前述の、判定条件／条件網羅で示したデータに欠陥がある。例えば，実際にコーディングされた結果が次のように「ＯＲ」でなく「ＡＮＤ」と成っていても次の誤りを検出できない。

ＩＦ　Ａ　AND　ＴＨＥＮ　Ｓ；

複数条件網羅では，判定条件の中の個々の条件の結果の全ての組合せ少なくとも１回は確かめるようにテストデータを作成する。すなわち
Ａ：「真」，Ｂ：「真」
Ａ：「真」，Ｂ：「偽」
Ａ：「偽」，Ｂ：「真」
Ａ：「偽」，Ｂ：「偽」
となるように，４つのテストデータを作成する。

付録４

メトリックス例（ISO/IEC9126対応）-1/6

１．機能性（functionality）

― 機能の集合の存在及びそれらの明示された性質の存在をもたらす属性の集合

機能は、明示的又は暗示的な必要性を満たすものとする

- ○ 合目的性；ユーザ改良件数、仕様の改定率
- ○ 正確性；ユーザから要求された数値精度の実現率、説明書と実動作の合致度
- ○ 相互運用性；データ形式の合致度、インターフェースやプロトコルの合致度
- ○ セキュリティ；暗号化率、アクセス履歴保有率
- ○ 機能性標準適合性；関連する全ての規格中、遵守している項目の割合

付録４

メトリックス例（ISO/IEC9126対応）-2/6

２．信頼性（reliability）

― 明示された条件の下で、明示された期間、ソフトウェアの達成のレベルを維持するソフトウェアの能力をもたらす属性の集合

- ○ 成熟性；平均故障間隔（ＭＴＢＦ）、平均故障寿命（ＭＴＴＦ）
- ○ 障害許容性；誤入力．誤操作検出力、単位障害あたりのダウン回数
- ○ 回復性；平均修復時間（MTTR）、平均ダウンタイム（MDT）
- ○ 信頼性標準適合性；関連する全ての規格中、遵守している項目の割合

付録４

メトリックス例（ISO/IEC9126対応）-3/6

３．使用性（usability）

― 明示的又は暗示的な利用者の集合が、使用するために必要とする労力及び個々の使用結果による評価に影響する属性の集合

- ○ 理解性；デモが用意されている機能の数
- ○ 習得性；マニュアル装備率、学習機能装備率
- ○ 運用性；メッセージ的確度、キータッチ数、レスポンス時間
- ○ 魅力性；利用者にとって魅力的である機能の数
- ○ 信頼性標準適合性；関連する全ての規格中、遵守している項目の割合

付録４

メトリックス例（ISO/IEC9126対応）-4/6

４．効率性（efficiency）

― 明示的な条件の下で、ソフトウェアの達成のレベルと使用する資源の量との間の関係に影響する属性の集合

- ○ 時間効率性；処理速度（CPU実行時間、入出力処理時間など）、処理能力（トランザクション処理件数など）
- ○ 資源効率性；資源使用量（CPU使用量、メモリ使用量など）、資源使用率（単位時間当たりの、CPU利用率、メモリ利用率など）
- ○ 効率性標準適合性；関連する全ての規格中、遵守している項目の割合

付録４

メトリックス例（ISO/IEC9126対応）-5/6

５．保守性（maintainability）

ー　仕様化された改定を行うために必要な労力に影響する属性の集合

- ○　解析性；誤り箇所識別率、診断機能の装備率
- ○　変更性；ＫＬＯＣあたりの修正実施時間、
  障害１件あたりの修正実施時間
- ○　安定性；モジュールの結合度
- ○　試験性；KLPCあたりのテスト時間、障害１件あたりのテスト時間
- ○　保守性標準適合性；関連する全ての規格中、遵守している
  項目の割合

付録４

メトリックス例（ISO/IEC9126対応）-6/6

６．保守性（portability）

ー　ソフトウェアのある環境から他の環境へ移す際の、そのソフトウェアの
能力をもたらす属性の集合

- ○　環境適応性；適用可能機種／ＯＳ数、変更せずに使用できる
  データの割合
- ○　設置性；ソースコードの変更率、移行ツールの適用できる割合
- ○　共存性；共通資源を他のソフトウェアと共存できる割合
- ○　置換性；置換前のソフトウェアと比べて、実現できる機能や
  性能の割合
- ○　保守性標準適合性；関連する全ての規格中、遵守している
  項目の割合