# Software Design
# 10. Verification & Validation

Natsuko Noda
nnoda@shibaura-it.ac.jp




# ソフトウェア設計論
# 10. 検証 & 妥当性確認

野田 夏子
nnoda@shibaura-it.ac.jp




## Today's Topics

- Verification & validation
  - Techniques for checking software
  - Testing
  - Review




## 本日のお題

- 検証と妥当性確認
  - ソフトウェアを確認するための技術
  - テスト
  - レビュー

# Verification & Validation




# 検証と妥当性確認




## Checking Artifacts

- Artifacts
  - programs, data, documents and others that are developed in the software development process.
  - not only programs and manuals that are final development goals, but also specification documents, design documents, test data and others that are developed and used in the development process.
- We must check all those artifacts.
  → Verification and validation (V&V)




## 成果物の確認

- 成果物
  - ソフトウェア開発で作られるプログラム、データ、ドキュメント等
  - 最終的な開発目標であるプログラムやマニュアルだけでなく、開発過程で作られる仕様書、設計書、テストデータなども含まれる
- これらすべての成果物を確認しなければならない
  → 評価と妥当性確認 (V&V)

cf. SE-J, 7.1

# Verification

- Activities to assess whether artifacts developed during a phase of the software development process meet the conditions and constraints imposed at the start of that phase.
- *"Are we building the product right?"*
- Ex. To check the design documents (developed in the design phase) meet the requirement specification (given  at the start of the design phase)
- To Confirm the consistency between what is given and what is developed.




# 検証

- 開発のあるフェーズで作られた成果物が、そのフェーズ開始時点に課せられた条件や制約を満たしているかどうかを評価する活動
- 「正しく作っているか」
- 例：(設計フェーズで作られた)設計書が(設計フェーズ開始時点に課せられた)要求仕様を満たしているかどうかを評価
- 与えられたものと作られたものとの突合せ作業

cf. SE-J, 7.1.1




# Validation

- to confirm that the requirements for the intended use and purpose have been achieved based on objective evidence
- *"Are we building the right product?"*
- Ex. When the goal is to know sales immediately, to find out how well this goal is achieved by the developed software, based on time to know sales data
- Confirmation of the achievement of real-world objectives




# 妥当性確認

- 意図される利用方法や目的に対する要求が達成されているかどうかを、客観的根拠を用意して確認すること
- 「正しいものを作っているか」
- 例：売上げを即座に把握したいという目的があった時に、構築されたソフトウェアによってそれがどの程度達成されるのかを、売上げ把握にかかる時間データに基づいて確認
- 現実世界の目的達成の確認

cf. SE-J, 7.1.1

# V&V

- Verification and Validation: V&V
- To check the artifacts, both of verification and validation are needed and important.
- These two activities, verification and validation, are sometimes referred to together as V&V.



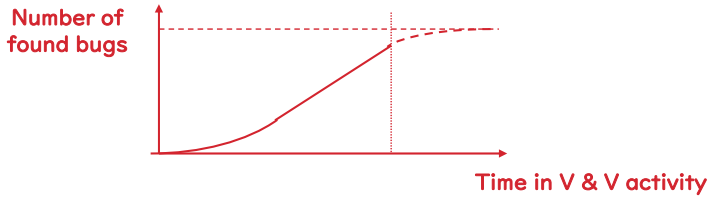

# V&V

- Verification and Validation: V&V
- 成果物の確認をするためには、検証と妥当性確認の両方が必要であり、重要
- これらふたつの活動、つまり検証と妥当性確認をまとめて、V&Vと呼ぶこともある




# Goal of V&V

- NOTE: we cannot confirm the software is developed completely "right".
  - aim of V & V is to establish confidence that the system is "fit for purpose".

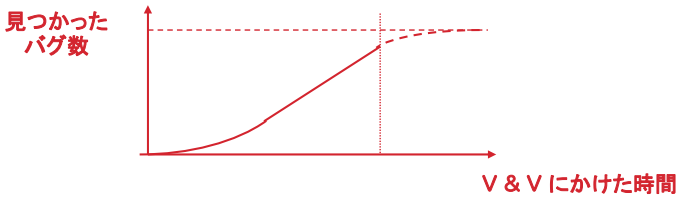






# V&Vの目的

- 注意: ソフトウェアが完全に「正しい」ことを確認することはできない
  - V & V の目的は、システムが「目的に合っている」ことをについて確信を持つこと



# Typical methods (1/4)

- Software testing
  - the system is executed with test data and its operational behaviour is observed.
  - Target: programs
  - we may find bugs.
  - NOTE :
    existence of bugs can be shown by finding bugs in testing, but existence of no bugs cannot be shown, even if we found no bugs in testing.




# 典型的な手法 (1/4)

- テスト
  - テストデータを使ってシステムを実行し、振舞いを観測すること
  - 対象: プログラム
  - バグを見つける
  - 注意：
    テストでバグを見つけることにより、バグの存在を示すことができるが、テストでバグが見つからなかったからといってバグが存在しないことを示すことはできない

cf. SE-J, 7.1.2, 7.2




# Typical methods (2/4)

- Review
  - we discuss the evaluation of outputs from each phase and whether we can accept outputs from the previous phase.
  - Target: specification, design, code, …
  - various stakeholders (contractee, manager, users, developers, etc.) participate, as needed.




# 典型的な手法 (2/4)

- レビュー
  - 各フェーズからのアウトプットを評価したり、前のフェーズからのアウトプットを受け入れられるかどうかを議論したりする
  - 対象: 仕様、設計、コード …
  - 様々なステークホルダ (契約者、管理者、ユーザ、開発者等)が必要に応じて参加

cf. SE-J, 7.1.2

# Typical methods (3/4)

- Static analysis
  - analyzes artifacts without executing them, using tools.
  - Examples
    - Analysis of program structure
    - check of compliance with coding rules
    - measurement of software metrics




# 典型的な手法 (3/4)

- 静的解析
  - ツールを使って成果物を実行せずに解析する
  - 例
    - プログラム構造の解析
    - コーディング規約通りであることの確認
    - ソフトウェアメトリクスによる計測

cf. SE-J, 7.1.2




# Typical methods (4/4)

- Formal verification
  - To verify software using formal and mathematical methods.
    - Theorem proving
    - Model checking
  - Target: specification, design, program, …




# 典型的な手法 (4/4)

- 形式検証
  - 数理論理学に基づいてソフトウェアの検証を行う
    - 定理証明
    - モデル検査
  - 対象: 仕様、設計、プログラム …

cf. SE-J, 7.1.2

# Testing



# テスト

cf. SE-J, 7.2



## Purpose

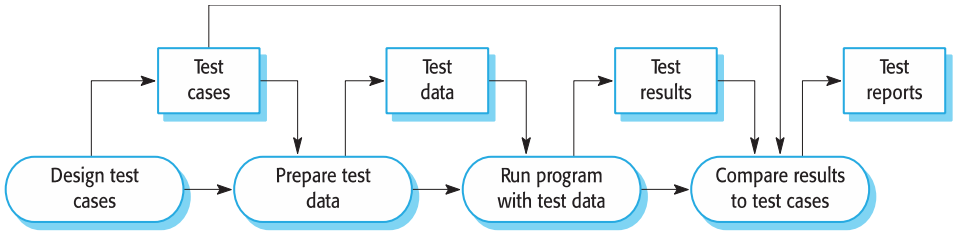
- To find bugs by executing the target software.
  - To check whether required functions the software has
  - To check whether qualities at run time such as response speed and reliability is as expected
  - to determine whether the development can be ended; whether the software can be released; etc.



## 目的

- 実際にソフトウェアを動作させて欠陥を発見すること
  - ソフトウェアが持つ機能を確認
  - 実行時の性能や信頼性が期待通りかを確認
  - 開発を終えて良いか、ソフトウェアをリリースして良いか等を決定

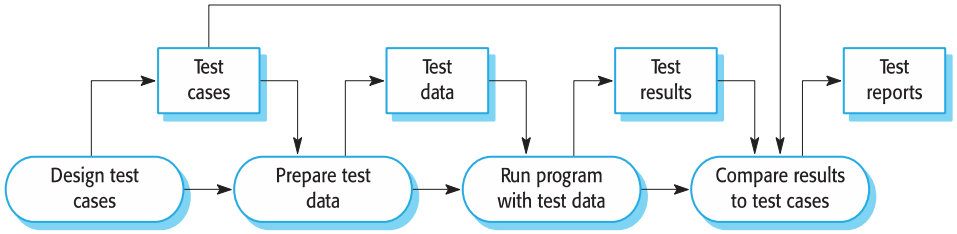
# A model of the software testing process




test case: a pair of precondition, input, expected output, and post-condition.




# ソフトウェアのテスト工程のモデル




test case: a pair of precondition, input, expected output, and post-condition.




# Test case and test suite

- Test case
  - A set of input values, execution conditions, and expected output values
- Test suite
  - A collection of test cases for the test target (the whole system, a sub-system, a component, etc.)




# テストケースとテストスイート

- テストケース
  - 入力値、実行条件、期待される出力値の集合
- テストスイート
  - テスト対象となるシステムやコンポーネントのためにテストケースをまとめたもの

# Test level

- Categorization of testing by its target and its purpose.
  - Unit testing, (or, component testing), where individual units, e.g. program units, object classes, and components, are tested. To separate a unit from other parts, drivers and stubs are needed.
    Done by a person who develops the target unit.
  - Integration testing, where several individual units are integrated, and interfaces and communication between them are tested. Drivers and stubs may be needed.
    Done by a person who integrates units, or a team specialized in testing.

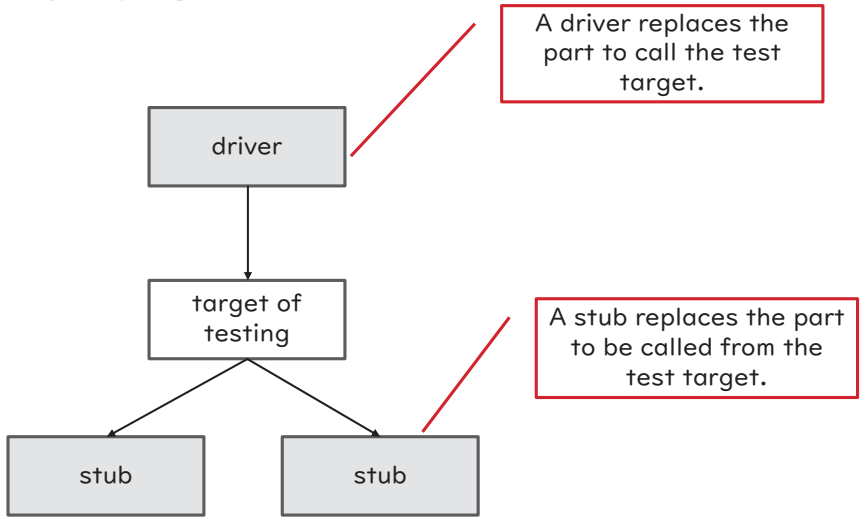



# テストレベル

- そのテストがどのような対象に対してどのような目的をもって行われるかという位置づけという観点からの分類
  - 単体テスト (もしくはコンポーネントテスト) : 特定のプログラム、オブジェクトクラス、コンポーネントのテスト。その部分を他の部分から切り離すために、ドライバとスタブが必要
    対象となる単体を開発した人によって行われる
  - 統合テスト ：単体を結合し、それらの間のインタフェースや相互処理をテスト。
    ドライバ、スタブが必要になることもある。
    統合の担当者、もしくはテストの専門チームが行う



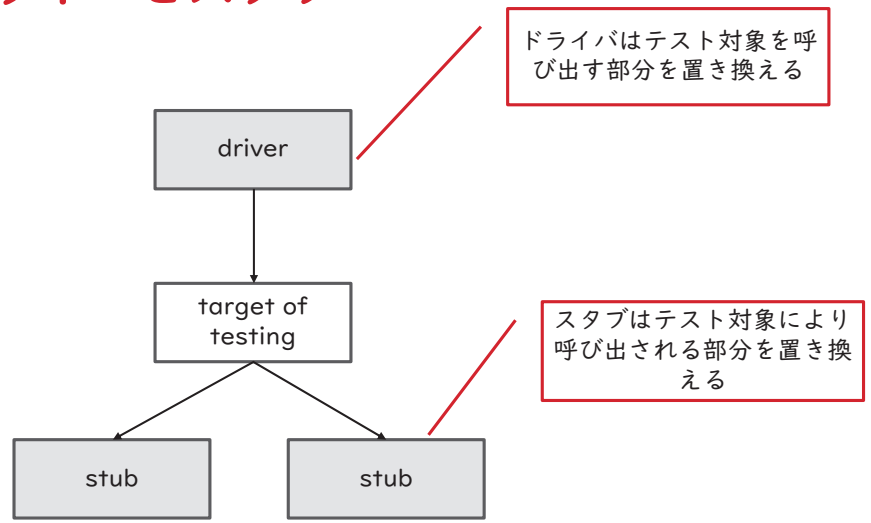

# Driver & Stub







# ドライバとスタブ



# Test level (Cont.)

- System testing, where the whole system is tested. The behavior of the system is tested and it it checked whether the behavior meets the functional and non-functional specification or not. Done by a team specialized in testing.
- Acceptance testing, where the customer determines whether the system achieves the purpose and is acceptable or not.




# テストレベル (Cont.)

- システムテスト ：システム全体をテストする。システム全体の振舞いがテストされ、機能や非機能に関して仕様通りであるかどうかを確認
  テスト専門チームにより行われる
- 受入れテスト ：顧客がシステムが目的を達成していて、受け入れる可能であるかどうか決めるために行われるテスト




# Test type

- Categorization of testing by its purpose
  - Functional testing: Testing to ensure that the system implements the functionality according to the specifications.
  - Non-functional testing: Testing about non-functional aspects. Various tests are included; performance testing, load testing, stress testing, usability testing, interoperability testing, reliability testing, portability testing.




# テストタイプ

- テストの目的による分類
  - 機能テスト：システムが仕様通りに機能を実装しているかどうかを確認するためのテスト
  - 非機能テスト：非機能的な側面を確認するテスト。性能テスト、負荷テスト、ストレステスト、ユーザビリティテスト、相互運用性テスト、保守性テスト、信頼性テスト、移植性テストなど様々なテストが含まれる

# Test type (Cont.)

- Structural testing: Testing based on the architecture of the system and the structure of its components. Testing to see what's going on internally.
- Testing when the software is changed: Testing after changes are made to ensure that defects are removed, or no new defects are created.
  - Conformance testing (re-testing): Testing to see if the defects, which were found by the last test, have been corrected correctly. A test that was failed last time is re-run to see if it passes.
  - Regression testing: Testing to make sure that the modifications have not created any other defects. A test that was passed last time is re-run to see if it is not failed due to a side effect of the modification.




# テストタイプ (Cont.)

- 構造テスト：システムのアーキテクチャやコンポーネントの構造に基づくテスト。内部で何が起こっているか確認するテスト
- 変更に伴うテスト：変更後に欠陥が除去されているか、新たな欠陥が作りこまれていないかを確認するテスト
  - 確認テスト(再テスト)：テストにより発見された欠陥を修正した後に、その欠陥が正しく修正されているかどうかを確認するためのテスト。前回不合格に終わったテストを再度実行して、合格するかどうかを確認するもの
  - 回帰テスト：修正によって欠陥を作りこんでいないかを確認するためのテスト。前回合格だったテストが、修正の副作用で不合格になっていないかどうかを確認するもの




# Black-box testing

- **is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.**
  - Specification-based testing




# ブラックボックステスト

- **システムやコンポーネントの内部機構を無視して、選択した入力と実行条件に応じて生成される出力のみに焦点を当てるテスト**
  - 仕様ベースのテストとも呼ばれる

# Strategies for black box testing

- Equivalence partitioning
- Boundary value analysis
- Decision table testing
- State transition testing
- Use case testing




# ブラックボックステストの技法

- 同値分割
- 境界値分析
- 決定表テスト
- 状態遷移テスト
- ユースケーステスト

cf. SE-J, 7.2.5




# Equivalence partitioning

- Divides the input to equivalence partition
  - divides the input domain of a program into classes. For each of these equivalence classes, the set of data should be treated the same by the module under test and should produce the same answer. Test cases should be designed so the inputs lie within these equivalence classes.

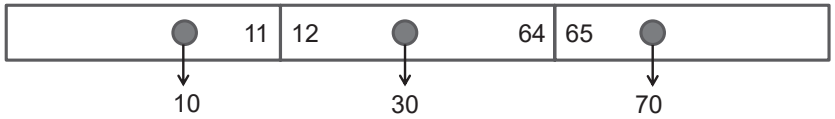



# 同値分割

- 入力を同値クラスに分割
  - プログラムの入力値を同値クラス(同値領域)に分割。これら同値クラスのそれぞれは、データのセットは、テスト対象のモジュールによって同じように扱われるデータのセットで、同じ答えを生成しなければならない。
    テストケースは、入力がそれぞれこれらの同値クラス内に入るように設計

# Equivalence partitioning

- Ex. To test the program "child fee if under 12 years old, senior fee if over 65 years old, adult fee otherwise."
  - Equivalence partitions are "under 12 years old," "between 12 and 65 years old," and "65 years and over."
  - Test with for example 10 years old, 30 years old, and 70 years old.

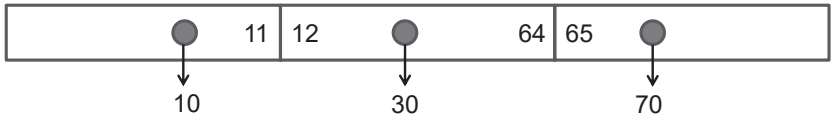






# 同値分割

- 例.「12歳未満は子供料金、65歳以上はシニア料金、それ以外は大人料金」とするプログラムをテスト
  - 同値クラスは 「12歳未満」 「12歳以上65歳未満」 「65歳以上」の3つ
  - この場合に、例えば、10歳、30歳、70歳をテスト

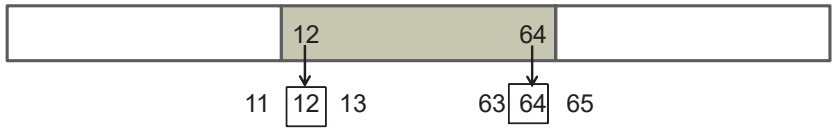






# Boundary value analysis

- to focus testing at the "boundaries."
  - programmers often make mistakes on the boundaries of the equivalence classes/input domain.
- Same ex. on the previous page



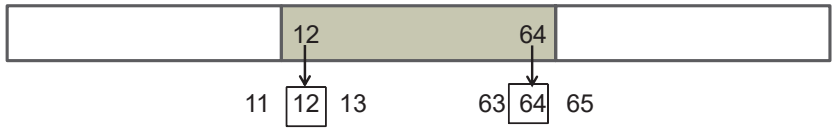

Test boundary values (12 and 64).
Also test neighborhoods that are not in the partition; 11 an 65.

In other approach, both neighborhoods of boundary values are tested; 11 and 13, and 63 and 65.




# 境界値分析

- 「境界」に着目したテスト
  - 同値クラスや入力ドメインの境界で間違いを起こしやすい
- 前頁と同じ例題




境界値(12と64)をテスト。
また、パーティションに含まれていない境界値の近傍もテスト。

境界値の両方の近傍をテストする方法もある。
この場合、11と13、63と65

# Decision table testing

- Decision tables are used to record complex business rules that must be implemented in the program, and therefore tested.
  - Decision tables shows the correspondence between the combination of an input and conditions and outputs.




# 決定表テスト

- 決定表は、プログラムに実装しなければならない複雑なビジネスルールを記録するために使用され、それをテストする
  - 決定表は入力と条件と出力の組み合わせの対応を示す




# Decision table example

| | Rule 1 | Rule 2 | Rule 3 |
|---|---|---|---|
| Conditions | | | |
| A lands on B's property | Yes | Yes | No |
| A has enough money to pay rent | Yes | No | -- |
| Actions | | | |
| A stays in game | Yes | No | Yes |

Test all rules.




# 決定表の例

| | Rule 1 | Rule 2 | Rule 3 |
|---|---|---|---|
| 条件 | | | |
| AがBの土地に上陸 | Yes | Yes | No |
| Aは家賃を払うだけのお金あり | Yes | No | -- |
| アクション | | | |
| Aはゲーム内にとどまる | Yes | No | Yes |

全てのルールをテストする

# Other strategies

- State transition testing
  - to design testing based on state transition diagram (state machine diagram) or state transition table.
  - There are some methods to design test cases from state transition diagram (table).
- Use case testing
  - to design testing based on use case description.
  - use pre-condition, post-condition, main sequence, alternative sequence, etc.




# 他の方法

- 状態遷移テスト
  - 状態遷移図（ステートマシン図）や状態遷移表を使ってテストを設計
  - 状態遷移図（表）からテストケースを設計するいくつかの方法がある
- ユースケーステスト
  - ユースケースの記述に基づいてテストを設計
  - 事前条件、事後条件、基本系列主、代替系列などを使用




# White-box testing

- is testing that takes into account the internal mechanism of a system or component (IEEE, 1990).
  - Including control flow testing, data flow testing, and others.




# ホワイトボックステスト

- システムやコンポーネントの内部機構を考慮したテストである（IEEE, 1990）
  - 制御フローテスト、データフローテストなど

# Coverage

- Coverage is a measure of the completeness of the set of test cases in white-box testing.
- Statement coverage (C0 coverage)
  - a measure of the percentage of statements that have been executed by test cases.
- Branch coverage (C1 coverage)
  - a measure of the percentage of the decision points (Boolean expressions) of the program have been evaluated as both true and false in test cases.




# 網羅率

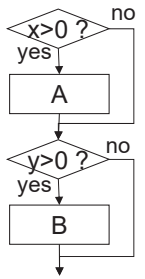
- ホワイトボックステストにおけるテストケースのセットの完全性の尺度
- 命令網羅 (C0 coverage)
  - テストケースで実行された命令の割合
- 分岐網羅 (C1 coverage)
  - ブール値で表現される分岐のうち、実行された分岐の割合

cf. SE-J, 7.2.6




# Example: Coverage

```
if (x>0) A;
if (y>0) B;
```

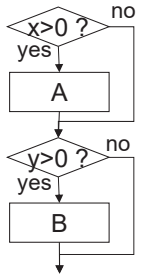



Statement coverage: 1 test case is needed.
{x>0, y>0}
Branch coverage: 2 test cases are needed.
{x>0, y>0}, {x<=0, y<=0}




# 例: 網羅率

```
if (x>0) A;
if (y>0) B;
```




命令網羅: 1つのテストケースが必要
{x>0, y>0}
分岐網羅: 2つのテストケースが必要
{x>0, y>0}, {x<=0, y<=0}

# Key points of testing

- Independence of testing
  - If the testing is conducted by people and teams who are not involved in development, the test is said highly-independent. This is desirable.
- Monitoring of testing
  - We cannot verify all the behavior in testing. We must terminate it in practical cost and time.
  - We need to monitor the status of testing.
  - use graphs to depict cumulative number of errors.
    - reliability growth curve



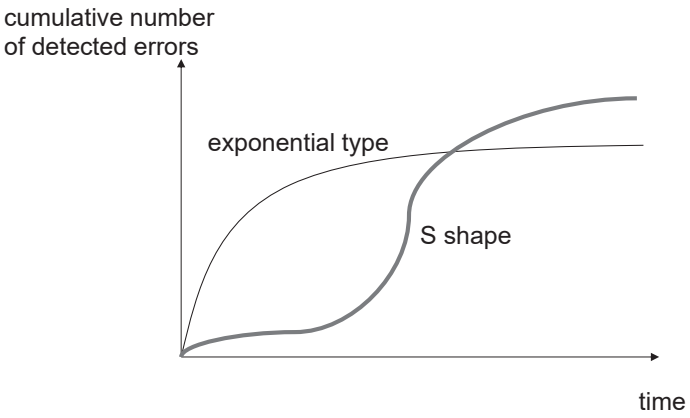

# テストの留意点

- テストの独立性
  - 開発に携わっていない人やチームがテストを行っている場合は、独立性が高いと言われる。これが望ましい形
- テストのモニタリング
  - テストで全ての動作を検証することはできない。実質的なコストと時間内に終了させなければならない
  - テストの状況を把握する必要
  - グラフを使用して累積エラー数を表現
    - 信頼度成長曲線

cf. SE-J, 7.3



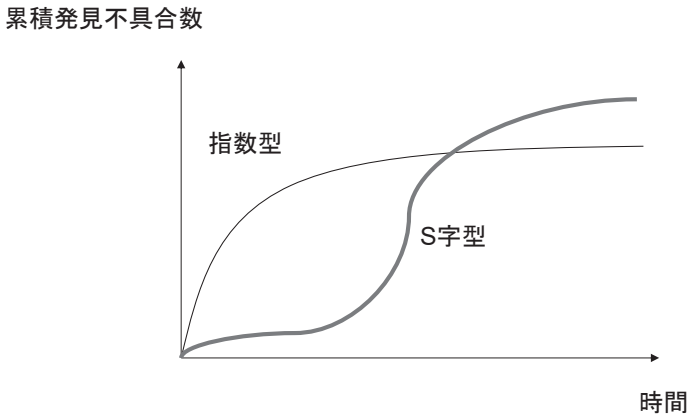

# Reliability growth curve







# 信頼度成長曲線



# Key points of testing (Cont.)

- Usage of tools
  - To automate testing is beneficial; there are many tools to automate testing.
  - Tool functionality: register test suites, execute testing using test data based on the test suites, and check the test results.




# テストの留意点 (Cont.)

- ツールの使用
  - テストを自動化することは有益であり、テストを自動化するためのツールがある
  - ツールが持つ機能: テストスイートの登録、テストスイートに基づくテストデータを使ったテストの実行、テスト結果のチェック




# Review




# レビュー

cf. SE-J, 7.2

# Purpose of Review

- is to detect defects in the review target by the members participating in the review.
  - Not suitable for rigorous and exhaustive checks
  - Confirmation based on empirical awareness by people
  - can be used in various development phases




# レビューの目的

- レビューに参加するメンバによって、レビュー対象の欠陥を検出すること
  - 厳密で網羅的な確認には不向き
  - 人による経験的な気づきに基づく確認ができる
  - 様々な工程で活用できる




# Review Type

- Management review：performed by a manager or someone in that position. Systematic evaluation of a software product or process.
- Technical review：to examine whether it is suitable for its intended use and whether it deviates from specifications and standards; performed by members those who have the ability to consider the issues. Systematic evaluation of software products.




# レビューの種類

- 管理者レビュー：管理者あるいはその立場の人によって実行される。ソフトウェア製品やプロセスの体系だった評価
- 技術レビュー：意図した利用に適しているか、仕様・標準から逸脱していないかなどを検討する資質を持ったメンバによって行う。ソフトウェア製品の体系だった評価

## Review Type (Cont.)

- Inspection：Visual inspection of software products to detect and identify defects such as errors and deviations from specifications and standards
- Walk through： A static analysis technique in which designers or programmers describe a software product and participants ask questions and comment on defects, development standard violations, and other issues.
- Audit：conducted by a third party to assess compliance with specifications, standards, contractual agreements, and other criteria. Independent Inspection.




## レビューの種類 (Cont.)

- インスペクション：誤りや仕様・標準からの逸脱などの欠陥を検出し識別するためのソフトウェア製品の目視検査
- ウォークスルー：設計者やプログラマがソフトウェア製品について説明し、参加者が欠陥、開発標準違反、その他の問題について質問やコメントすることによって行われる静的分析手法
- 監査：第三者によって実施され、仕様、標準、契約上の合意、その他の基準への準拠を査定する。独立した検査




## For your review

1. What is the difference between software verification and validation?
2. What is the white-box testing?
3. What is the similarity between testing and review? What is the difference?




## For your review

- (日本語訳省略)