中山大學 資管所
專題演講

從CMMI評鑑談軟體測試的涵蓋率

Testing Coverage Investigation in the aspects of CMMI Appraisal

2009/05/07 (星期四)

張文貴教授
長榮大學
資訊暨工程學院院長
wkc@thu.edu.tw

從CMMI評鑑談軟體測試的涵蓋率

大 綱

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- 使用模式(Usage Model)
- 結語

簡歷

- 現任
  - ✓長榮大學 資訊暨工程學院 院長
  - ✓東海大學 資訊工程與科學系 教授
  - ✓中華民國品質學會 理事長
  - ✓台灣軟體流程改善聯盟(SPIN-Taiwan) 監事
  - ✓SEI授權CMMI-DEV & ACQ 講師, 主評鑑員
  - ✓標準檢驗局「資訊及通信國家標準技術委員會」委員
- 專長
  - ✓軟體品質管理、軟體流程改善、軟體專案管理、軟體工程、軟體測試、軟體可靠度

如何分辨偽鈔

- 英國銀行協會每年都會舉辦訓練班，教導銀行職員如何分辨偽鈔
- 有一次，請來一位鑑識專家為學員們講習。他是鈔票印製廠的工人，盯著鈔票印製已超過十年。
- 這名專家鑑別偽鈔的本領，讓所有參加訓練的銀行員都非常佩服。他只須注視一會兒，或輕輕一摸，就能立刻辨識出真假。
- 一名行員問他:
  - ✓「你研究偽鈔多久了？為什麼一眼就可以看出真偽？我們經常比對半天，都還分不出真假呢。」
- 鑑識專家回答:
  - ✓「我們並不研究偽鈔啊，印製廠裏接觸的都是真鈔，我們**只研究真鈔**！」

從CMMI評鑑談軟體測試的涵蓋率

大 綱

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- 使用模式(Usage Model)
- 結語

能力成熟度整合模式CMMI

- Capability Maturity Model Integration
- 是一個針對**產品**與**服務**發展的流程改善成熟度模式
  - ✓包含發展與維護活動的最佳執行方法
  - ✓涵蓋產品從起始到交付與維護的生命週期

五個成熟度等級

- 1. 初始級, Initial
- 2. 管理級, Managed
- 3. 定義級, Defined
- 4. 量化管理級, Quantitatively Managed
- 5. 最佳化級, Optimizing

CMMI® Model V1.2

	流程管理	專案管理	工程	支援
ML 5 **最佳化**	**組織創新與推展** **Organizational Innovation and Deployment**			**原因分析與解決方案** **Causal Analysis and Resolution**
ML 4 **定量管理**	**組織流程績效** **Organizational Process Performance**	**量化專案管理** **Quantitative Project Management**		
ML 3 **定性管理**	**組織流程專注** **Organizational Process Focus** **組織流程定義 + *IPPD*** **Organizational Process Definition + IPPD** **組織訓練** **Organizational Training**	**整合專案管理 + *IPPD*** **Integrated Project Management + IPPD** **風險管理** **Risk Management**	**需求發展** **Requirements Development** **技術解決方案** **Technical Solution** **產品整合** **Product Integration** **驗證** **Verification** **確認** **Validation**	**決策分析與解決方案** **Decision Analysis and Resolution**
ML 2 **基本管理**		**專案規劃** **Project Planning** **專案監控** **Project Monitoring and Control** **供應商協議管理** **Supplier Agreement Management**	**需求管理** **Requirements Management**	**建構管理** **Configuration Management** **流程與產品品質保證** **Process and Product Quality Assurance** **度量與分析** **Measurement and Analysis**

從CMMI評鑑談軟體測試的涵蓋率

大 綱

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- → 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- 使用模式(Usage Model)
- 結語

Verification

to ensure that selected **work products** meet their specified requirements

➢SG 1 Prepare for Verification

- ✓SP 1.1 Select Work Products for Verification
- ✓SP 1.2 Establish the Verification Environment
- ✓SP 1.3 Establish Verification Procedures and Criteria

➢SG 2 Perform Peer Reviews

- ✓SP 2.1 Prepare for Peer Reviews
- ✓SP 2.2 Conduct Peer Reviews
- ✓SP 2.3 Analyze Peer Review Data

➢SG 3 Verify Selected Work Products

➢ ***Selected work products are verified against their specified requirements.***

- ✓SP 3.1 Perform Verification
- ✓SP 3.2 Analyze Verification Results

Appraisal Considerations for VER

- SG 1 Prepare for Verification
- SP 1.1 Select the work products to be verified and the verification methods that will be used for each.
  - ✓Work products are selected based on their contribution to meeting project objectives and requirements, and to addressing project risks.
  - ✓Methods of verification include, but are not limited to, inspections, peer reviews, audits, walkthroughs, analyses, simulations, testing, and demonstrations.

For Software Engineering

- Examples of verification methods include the following:
  - ✓Path coverage testing
  - ✓Load, stress, and performance testing
  - ✓Decision-table-based testing
  - ✓Functional decomposition-based testing
  - ✓Acceptance tests

binarySearch() Example

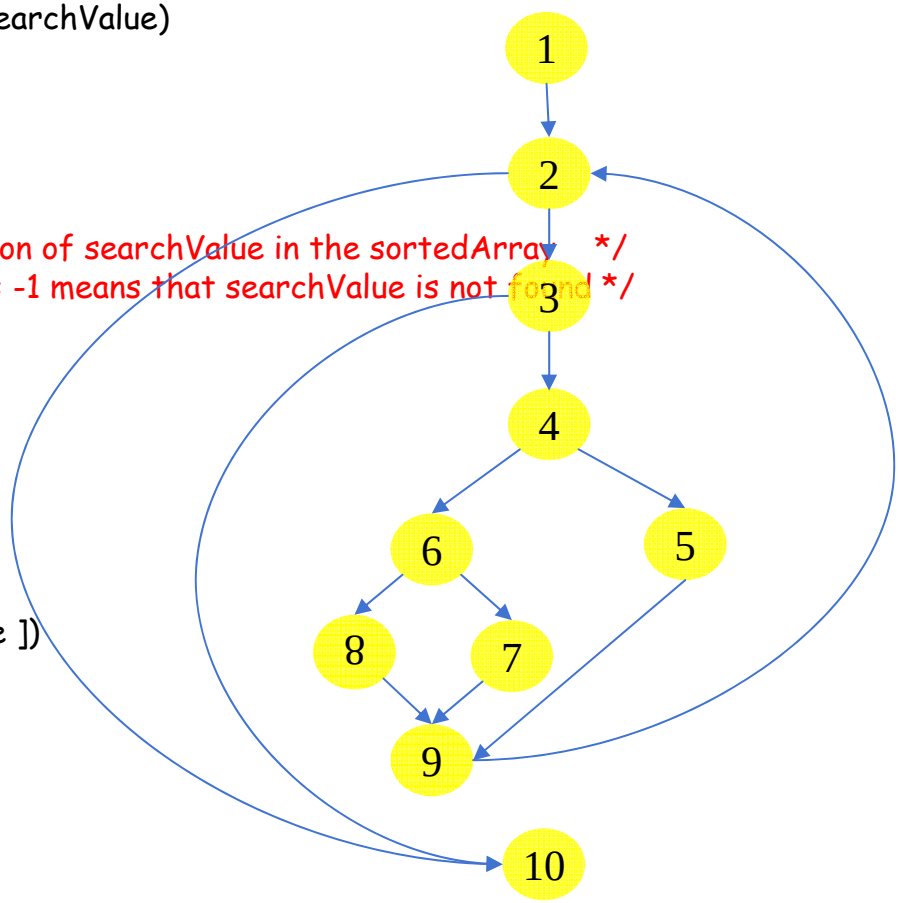
```
public int binarySearch(int sortedArray[ ], int searchValue)
{
    int bottom = 0;
    int top = sortedArray.length - 1;
    int middle, locationOfsearchValue;
    boolean found = flase;
    locationOfsearchValue = -1;

    while ( bottom <= top && !found)
    {
        middle = (top + bottom)/2;
        if (searchValue == sortedArray[ middle ])
        {
            found = true;
            locationOfsearchValue = middle;
        }
        else if (searchValue < sortedArray[ middle ])
                top = middle - 1;
        else
                bottom = middle + 1;
    } // end while

    return locationOfsearchValue;
}
```

1 2 3 4 5 6 7 8 9 10

/* the location of searchValue in the sortedArray */
/* location = -1 means that searchValue is not found */



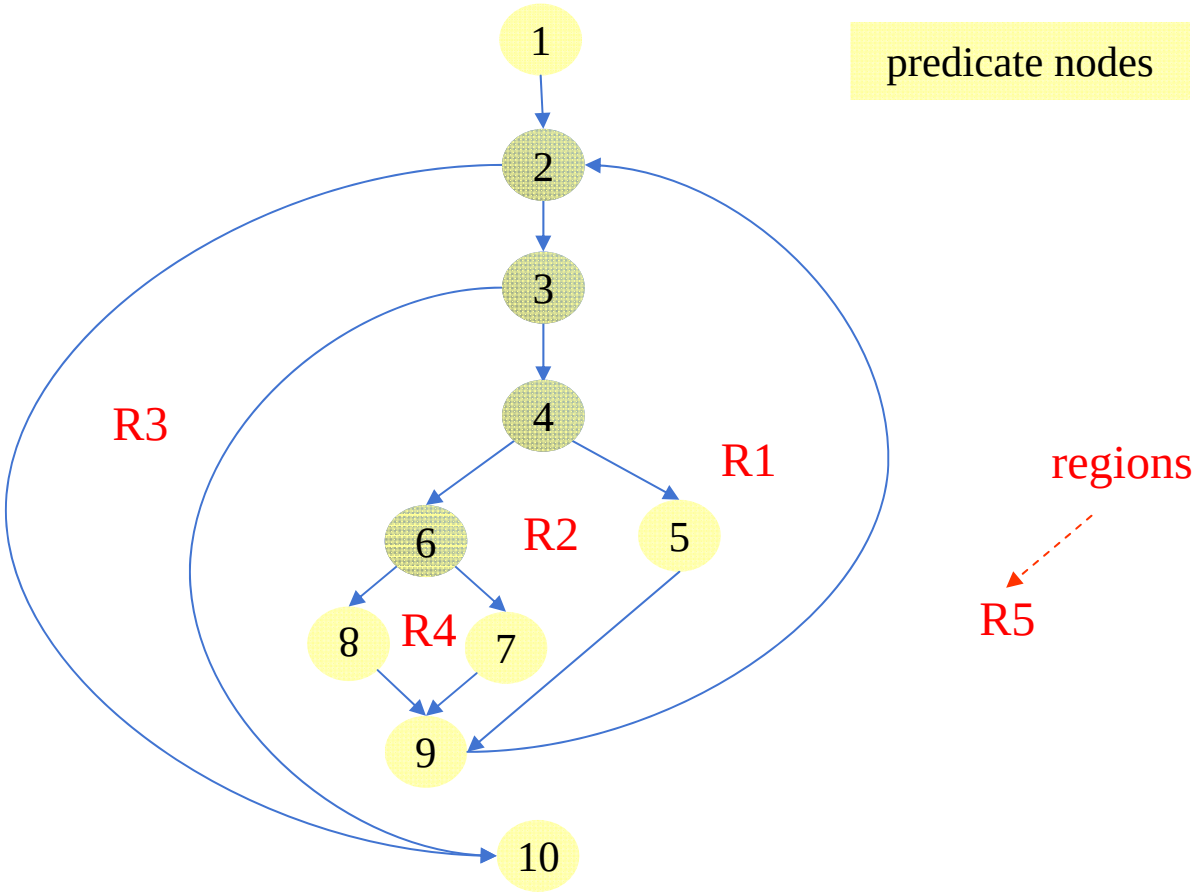
Basis Path Testing

- Flow graph notation (control flow graph)
  - ✓ Node represents one or more procedural statements.
    - ✶ **A sequence of process boxes and a decision diamond can map into a single node**
    - ✶ **A predicate node is a node with two or more edges emanating from it**
  - ✓ Edge (or link) represents flow of control
  - ✓ Region: areas bounded by edges and nodes
    - ✶ **When counting regions, include the area outside the graph as a region**

Cyclomatic Complexity

- Three ways to compute cyclomatic complexity:
  - ✓The number of regions of the flow graph correspond to the cyclomatic complexity.
  - ✓Cyclomatic complexity, V(G), for a flow graph G is defined as V(G) = E - N + 2
    where E is the number of flow graph edges and N is the number of flow graph nodes.
  - ✓Cyclomatic complexity, V(G) = P + 1
    where P is the number of predicate nodes contained in the flow graph G.

Cyclomatic Complexity of Function binarySearch()



For Software Engineering

- Examples of verification methods include the following:
  - ✓Path coverage testing
  - ✓Load, stress, and performance testing
  - ✓Decision-table-based testing
  - ✓Functional decomposition-based testing
  - ✓Acceptance tests

Decision-table-based Testing

- Consider an ATM system for providing withdrawal transaction service. The relevant conditions and actions of the system are:
  - ✓C1: The ATM card is valid
  - ✓C2: The password matches
  - ✓C3: There is enough money in the ATM machine
  - ✓A1: Dispense money
  - ✓A2: Prompt to indicate “not enough money”
  - ✓A3: Prompt to indicate “invalid ATM card or password”

Decision-table-based Testing (cont'd)

- The decision table is

	1	2	3	4
C1	F	T	T	T
C2		F	T	T
C3			T	F
A1			X	
A2				X
A3	X	X		

- **C: denotes a condition**
- **A: denotes an action**
- **T: denotes true**
- **F: denotes false**
- **X: denotes action to be taken.**

Decision-table-based Testing (cont'd)

- Action A1 is performed when conditions C1, C2, and C3 are true
- Action A2 is performed when conditions C1 and C2 are true and C3 is false
- Action A3 is performed when condition C1 is false or C2 is false

For Software Engineering

- Examples of verification methods include the following:
  - ✓Path coverage testing
  - ✓Load, stress, and performance testing
  - ✓Decision-table-based testing
  - ✓Functional decomposition-based testing
  - ✓Acceptance tests

normalize numeric expression

- The module reformats a numeric expression entered on a CRT.
  - ✓remove all commas, the sign, and the decimal point
  - ✓check the validity of the input expression
- Input
  - **✶A character string of length 25 called NUMERIC-EXPRESSION contains a numeric expression.**
  - **✶expression must contain at least 1 digit, may contain no more than 14 integer digits and no more than 4 fractional digits**
  - ✓Any of the following examples would be valid entries:

    +0
    1234.
    - .012
    12,345.
    6

normalize numeric expression
Module test case specification

- 3. Input specifications
  - ✓ **1,234. in NUMERIC-EXPRESSION**
- **4. Output specifications**

+12340000 in ALIGNED-NUMERIC-VALUE
NORMALIZATION-OK in RETURN-CODE
4 in INTEGER-DIGIT-COUNT
0 in FRACTIONAL-DIGIT-COUNT
N-0 in WAS-SIGN-FOUND
YES in WERE-COMMAS-FOUND
YES in WAS-DECIMAL-POINT-FOUND

normalize numeric expression Module test design specification

➢ 2. Features to be tested

Individual Features

2.1 Digits Only Processing
2.2 Sign Processing
2.3 Decimal Point Processing
2.4 Commas Processing

Combinations

2.5 Sign and Decimal Point
2.6 Sign and Commas
2.7 Decimal Point and Commas
2.8 Sign, Decimal Point and Commas

normalize numeric expression
Module test design specification

➢ 3. Approach refinements

✓Test case selection rationale.

- ✶Input constraints:
  - ↙(1) No more than 14 integer digits
  - ↙(2) No more than 4 fractional digits
  - ↙(3) No more than one decimal point
  - ↙(4) Between 1 and 3 contiguous digits to the left of each comma
  - ↙(5) Exactly 3 contiguous digits to the right of each comma
  - ↙(6) No commas after the decimal point

normalize numeric expression Module test design specification

➢ 4. Test identification

Digits Only		
Valid		
	14 integer digits	NNE.TC.001
	centered 6 integer digits	NNE.TC.002
	left justified 1 integer digit	NNE.TC.003
Invalid		
	15 integer digits	NNE.TC.010
	digit string with imbedded space	NNE.TC.011
	digit string with leading invalid character	NNE.TC.012
	digit string with imbedded invalid character	NNE.TC.013
	digit string with trailing invalid character	NNE.TC.014
Sign		
Valid		
	right justified + signed 14 integers	NNE.TC.020
	– signed integers	NNE.TC.021
Invalid		
	imbedded sign	NNE.TC.030
	trailing sign	NNE.TC.031
	sign alone without digits	NNE.TC.032
	2 leading signs	NNE.TC.033
	2 separated signs	NNE.TC.034
Decimal Point		
Valid		
	leading point with 4 fractional digits	NNE.TC.040
	embedded point with 1 fractional digit	NNE.TC.041
	trailing point with 14 integers	NNE.TC.042

normalize numeric expression Module test design specification

➢ 4. Test identification

Sign and Commas		
Valid		
	sign and comma with 14 digits	NNE.TC.100
	sign and comma with 4 digits	NNE.TC.101
Invalid		
	sign adjacent to comma	NNE.TC.110
Decimal Point and Commas		
Valid		
	comma with 14 integer digits and 4 fractional digits	NNE.TC.120
	one comma with 4 digits and trailing point	NNE.TC.121
Invalid		
	no digits between comma and point	NNE.TC.130
	4 digits between comma and point	NNE.TC.131
	comma following point	NNE.TC.132
Sign, Decimal Point, and Commas		
Valid		
	longest valid expression	NNE.TC.140
	shortest valid expression	NNE.TC.141
	representative valid expression	NNE.TC.142
Invalid		
	15 integer and 4 fractional digits	NNE.TC.150
	14 integer and 5 fractional digits	NNE.TC.151

normalize numeric expression
Module test case specification

➢ 1. Test case specification identifier

- ✓ NNE.TC**.121.01**
- ✓ One comma with 4 digits and trailing point.

➢ 2. Test items

- ✓ Normalized Numeric Expression Subroutine
  - ✶**This routine strips signs, commas, and decimal points from numeric expressions.**

➢ 3. Input specifications

- ✓ **1,234. in NUMERIC-EXPRESSION**

➢ 4. Output specifications

+12340000 in ALIGNED-NUMERIC-VALUE
NORMALIZATION-OK in RETURN-CODE
4 in INTEGER-DIGIT-COUNT
0 in FRACTIONAL-DIGIT-COUNT
N-0 in WAS-SIGN-FOUND
YES in WERE-COMMAS-FOUND
YES in WAS-DECIMAL-POINT-FOUND

Verification

to ensure that selected work products meet their specified requirements

➢SG 1 Prepare for Verification

- ✓SP 1.1 Select Work Products for Verification
- ✓SP 1.2 Establish the Verification Environment
- ✓SP 1.3 Establish Verification Procedures and Criteria

➢SG 2 Perform Peer Reviews

- ✓SP 2.1 Prepare for Peer Reviews
- ✓SP 2.2 Conduct Peer Reviews
- ✓SP 2.3 Analyze Peer Review Data

➢SG 3 Verify Selected Work Products

➢ ***Selected work products are verified against their specified requirements.***

- ✓SP 3.1 Perform Verification
- ✓SP 3.2 Analyze Verification Results

SP 1.3 Establish Verification Procedures and Criteria

- ***Establish and maintain verification procedures and criteria for the selected work products.***
- **Typical Work Products**
  - ✓1. Verification procedures
  - ✓2. Verification criteria
    - ✶**Verification criteria are defined to ensure that the work products meet their requirements.**

從CMMI評鑑談軟體測試的涵蓋率

大 綱

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- 使用模式(Usage Model)
- 結語

如何分辨偽鈔 1/2

- 英國銀行協會每年都會舉辦訓練班，教導銀行職員如何分辨偽鈔
- 有一次，請來一位鑑識專家為學員們講習。他是鈔票印製廠的工人，盯著鈔票印製已超過十年。
- 這名專家鑑別偽鈔的本領，讓所有參加訓練的銀行員都非常佩服。他只須注視一會兒，或輕輕一摸，就能立刻辨識出真假。
- 一名行員問他:
  - ✓「你研究偽鈔多久了？為什麼一眼就可以看出真偽？我們經常比對半天，都還分不出真假呢。」
- 鑑識專家回答:
  - ✓「我們並不研究偽鈔啊，印製廠裏接觸的都是真鈔，我們**只研究真鈔**！」

如何分辨偽鈔 2/2

- 鑑識專家繼續說:
- 「我們每天都要盯著真鈔好幾個小時，摸的也是真鈔，早已熟悉它的每個部分。而且在印製前，我們就必須仔細研究鈔票上所有的圖案，以及每項特色，還要背熟印製過程的每一個細節，這樣才能確保鈔票印出來的品質。盯真鈔久了，只要看到偽鈔，一眼就能看出哪個地方和我們熟悉的真鈔不同。」

Research Issues

- Test adequacy criteria
- Stopping rule determinates whether sufficient testing has been done
  - ✓How do you know when you have tested enough?
  - ✓How do you pick test cases?

Enhanced Coverage Indices

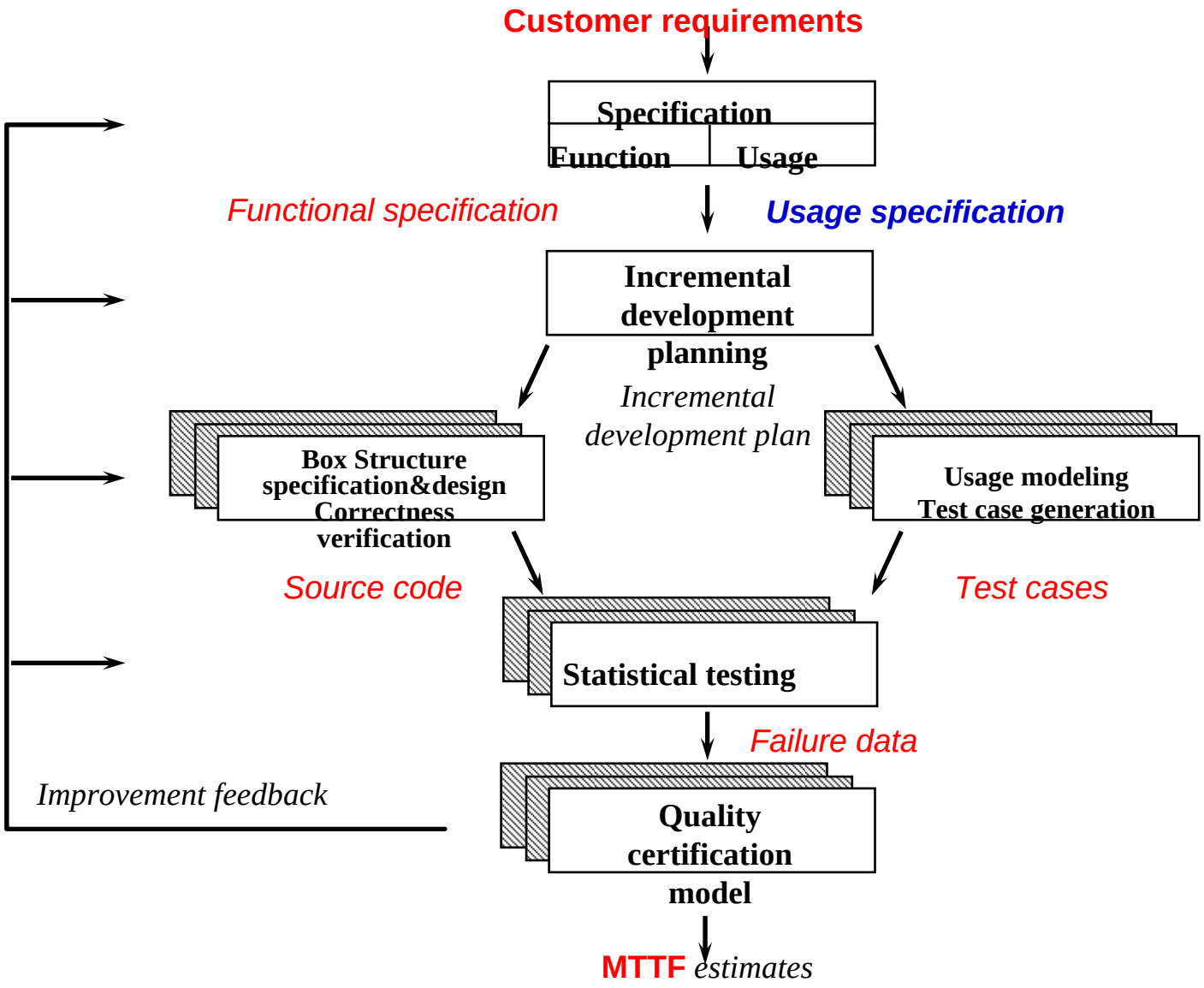
- Control flow coverage criteria
  - ✓Statement coverage
  - ✓Edge coverage
  - ✓Condition coverage
  - ✓Path coverage
- Functional coverage criteria
  - ✓Requirements coverage
  - ✓Use cases coverage
  - ✓Decision-table coverage

從CMMI評鑑談軟體測試的涵蓋率

大 綱

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- → 使用模式(Usage Model)
- 結語

zero-defect development—cleanroom approach

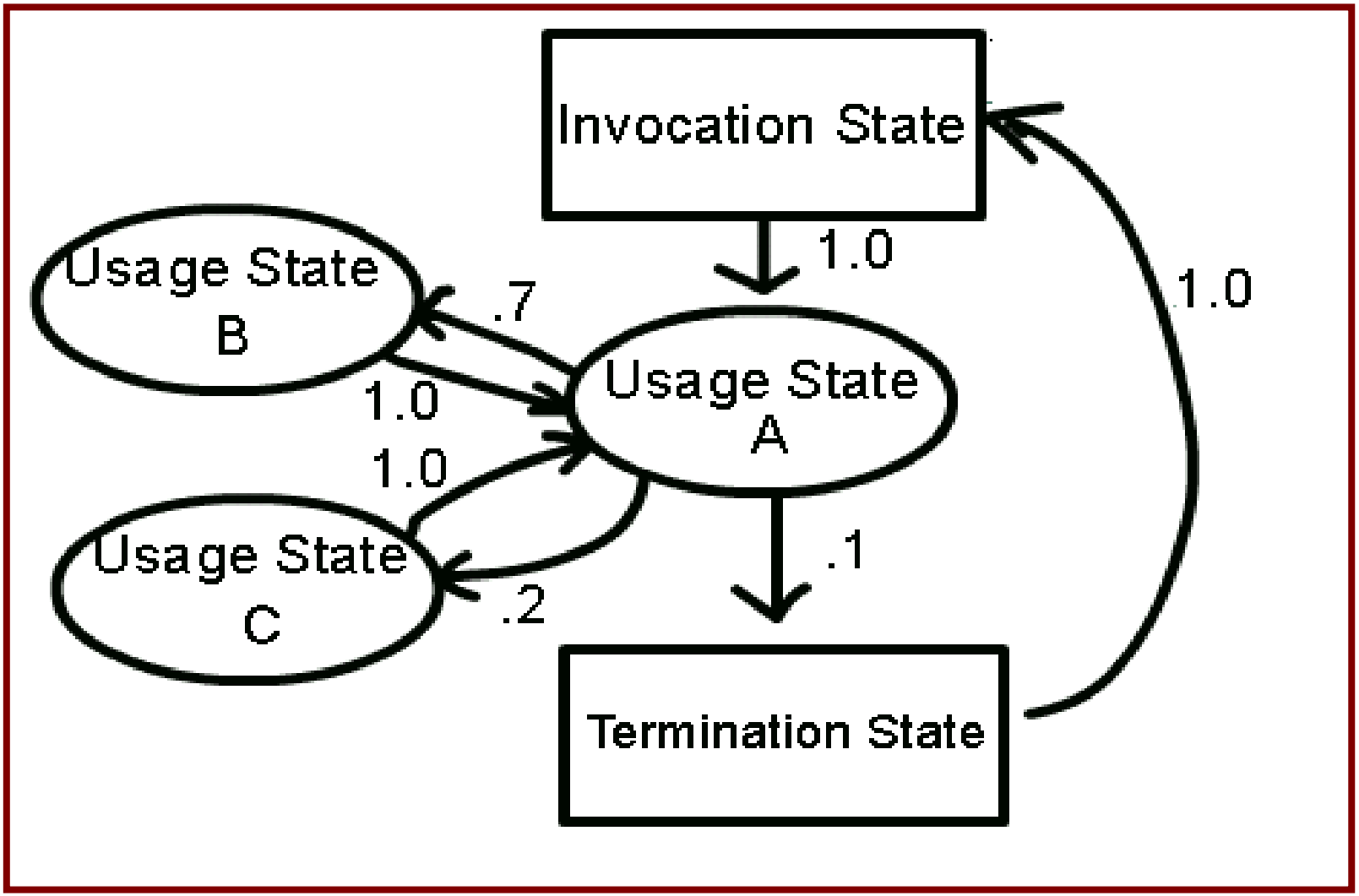


Windows 附屬應用程式—小算盤

小算盤 實際操作舉例

已鍵入	再鍵入	預期結果
CE 2 0 0	4	2004
CE 2 0 0 +	1 =	201
CE 2 0 0 +	+ =	400
CE 2 0 0 *	20 =	4000
CE 2 0 0 *	20 %	40.0

使用模式的簡單例子


Invocation State
1.0
1.0
Usage State B
.7
Usage State A
1.0
1.0
Usage State C
.2
.1
Termination State

使用模式(usage model)

- 特徵量化一個軟體系統的操作使用型態
  - ✓操作使用群體
    - ✶在預期環境中的預期軟體使用型態
    - ✶統計抽取測試案例
- 分析使用模式
  - ✓模式structure analysis
    - ✶**state coverage**
    - ✶**arc coverage**
    - ✶**no of test cases**
    - ✶**test script length**
  - ✓generate test script

Recent Research

- Practical stopping criteria for validating safety-critical software by estimating impartial reliability
  - ✓Applied Mathematical Modelling
  - ✓2007 July, 31-7 1411–1424

從CMMI評鑑談軟體測試的涵蓋率

大 綱

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- 使用模式(Usage Model)
- → 結語

結　語

- 軟體測試的涵蓋率
  - ✓Schedule
  - ✓Budget
  - ✓Risks
  - ✓Implemented techniques
  - ✓Reality trade-off
  - ✓Quality demands

William Shakespeare:
Time does not have the same appeal for everyone.

Quality does not have the same appeal for everyone.
“對於品質的感覺, 取決於您的態度!”

參考資料

- CMMI Product Team. *CMMI for Development, Version 1.2,* CMU/SEI-2006-TR-008, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006/08
- Practical stopping criteria for validating safety-critical software by estimating impartial reliability, Applied Mathematical Modelling, 2007/07, 31-7, 1411–1424
- 講義, 2009/02

從CMMI評鑑談軟體測試的涵蓋率

張文貴教授
wkc@thu.edu.tw

- 前言
  - ✓如何分辨偽鈔
- CMMI 的評鑑
- 驗證(Verification)流程領域的實施
- 研究議題
  - ✓軟體測試的涵蓋率
- 使用模式(Usage Model)
- 結語