軟體測試與品質管控

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

大綱

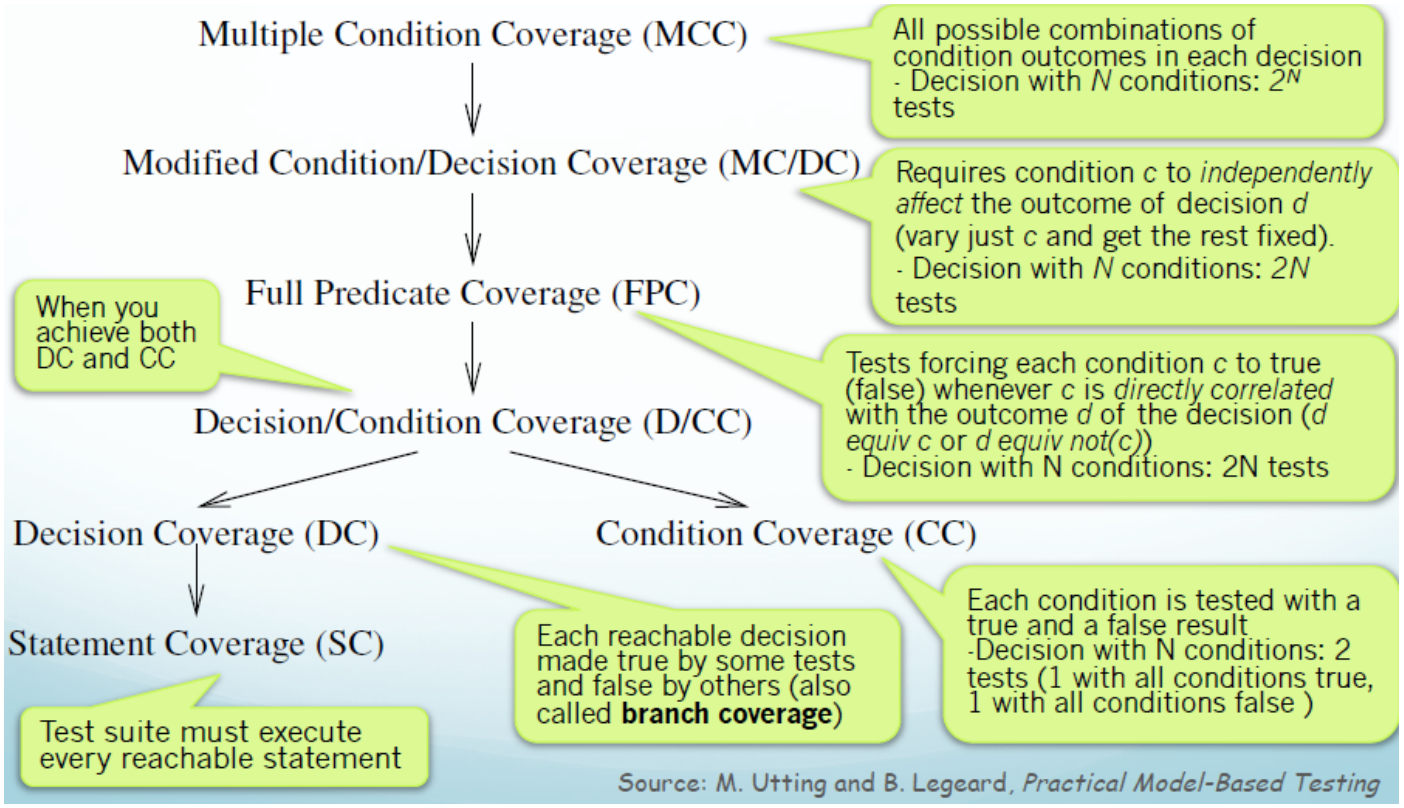
- 測試涵蓋度準則
- 靜態測試分析

測試涵蓋度準則

- 結構涵蓋度準則Structural coverage criteria/White-box testing
  - control-flow coverage
    - statement coverage
    - branch coverage/decision coverage/edge coverage
    - condition coverage
    - decision-condition coverage
    - multiple condition combination coverage
    - loop coverage
      - 通過0次，1次，多次
    - all path coverage

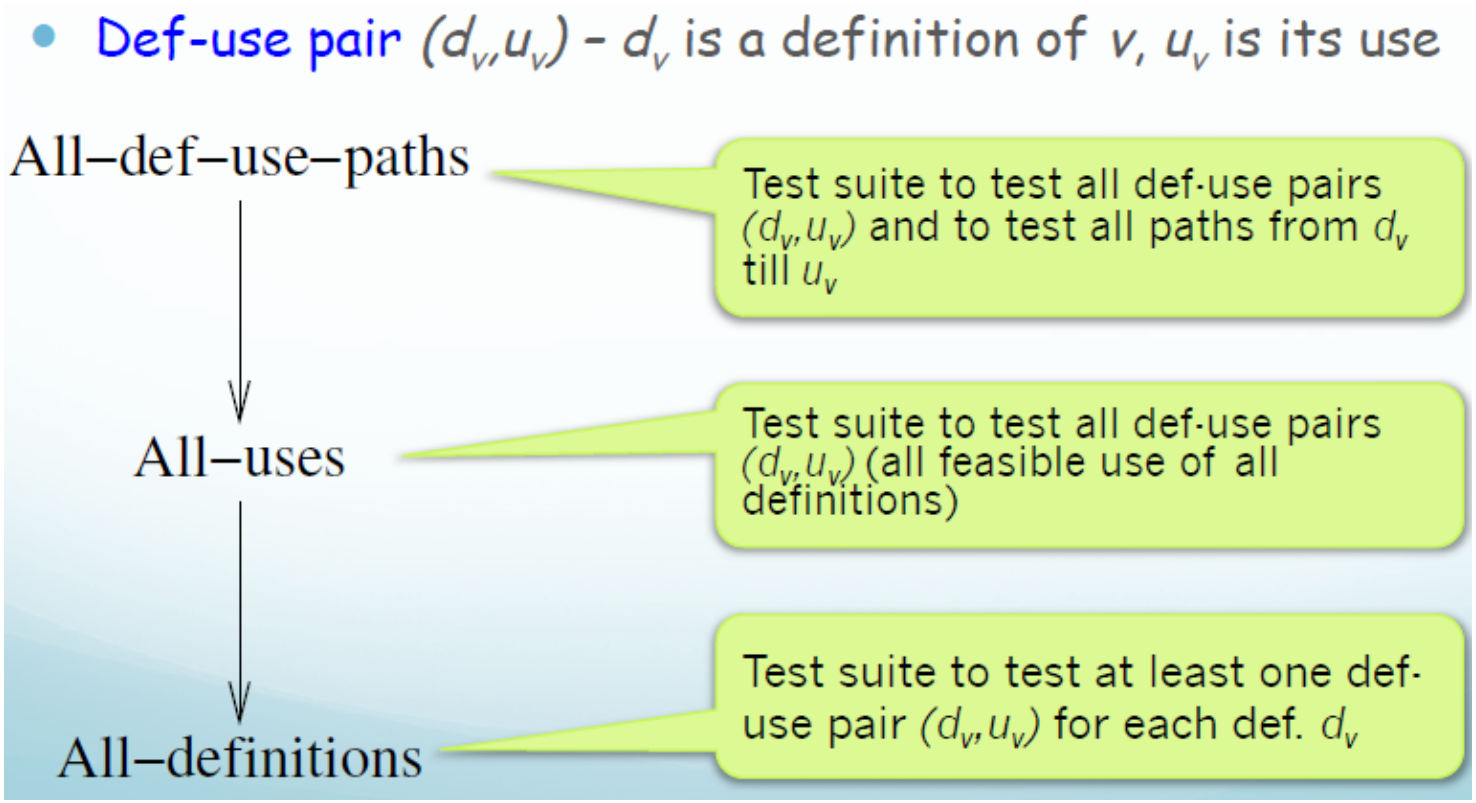
測試涵蓋度準則

- 結構涵蓋度準則Structural coverage criteria/White-box testing
  - control-flow coverage



測試涵蓋度準則

- 結構涵蓋度準則Structural coverage criteria/White-box testing
  - data-flow coverage

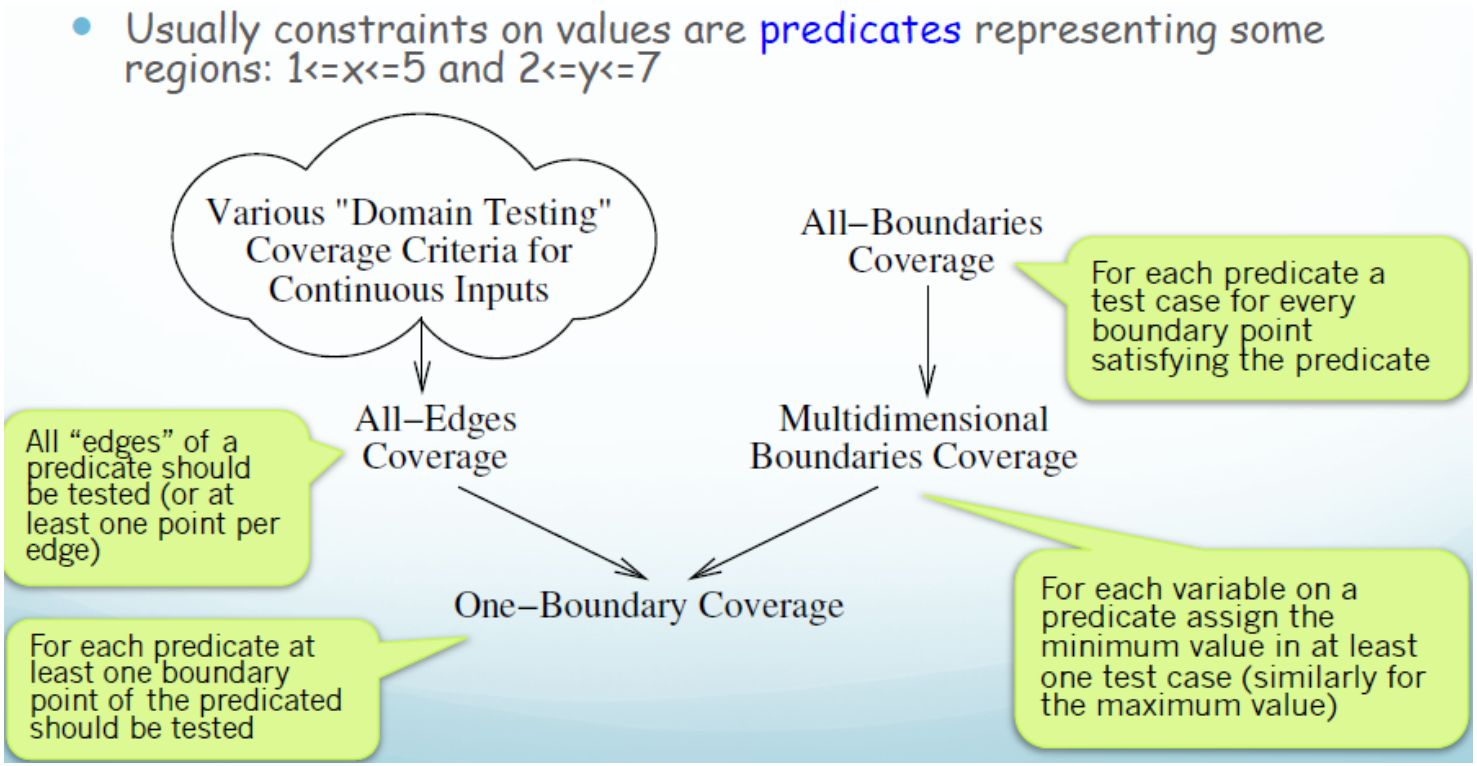


測試涵蓋度準則

- 資料涵蓋度準則Data coverage criteria/Black-box testing
  - 兩個極端coverage
    - one-value coverage: domain中取一個值
    - all-value coverage: domain中取所有的值
  - 實務上的coverage
    - statistical data coverage
      - random value coverage: random test
      - 使用Markov chains
    - pairwise testing
    - boundary coverage

測試涵蓋度準則

- 資料涵蓋度準則Data coverage criteria/Black-box testing
  - boundary coverage



測試涵蓋度準則

- 需求涵蓋度準則Requirements-based criteria
  - 每一個功能可測試的行為描述都要測試
- 錯誤涵蓋度準則Fault-model criteria
  - mutation testing

植錯(Fault seeding)

- 測量系統中尚有多少未被發現的錯誤
  - 若S為已知植入錯誤個數，U為系統中尚未被發現的錯誤個數，
  - 在一次測試中，總共發現F個錯誤，其中s個是先前植入的錯誤，則U=?

s: 測試發現到植入的錯誤 Discovered seeded fault,

S: 所有植入的錯誤 seeded fault

r: 測試發現到系統的錯誤 Discovered real fault,

R:系統所有的錯誤 Real fault

F: 測試發現所有的錯誤

U: 測試未發現的錯誤

F/U = s/S = r/R

靜態分析Static Analysis優點 1

- 降低專案整體成本。
  - 超過一半的軟體錯誤發生在需求或設計規格錯誤，審查與檢視可以在程式撰寫前發現錯誤，早期發現問題，可降低除錯與重測成本。

軟體發展階段	平均缺失產生率%	平均缺失排除的相對代價
需求規格	15%	1
設計	35%	2.5
單元編碼	30%	6.5
整合編碼	10%	16
文件撰寫	10%	40
系統測試	__	40
系統運作	__	110

靜態分析優點 2

- 靜態分析比動態測試有效率
  - 清楚發現錯誤所在及原因，不需經過繁複鑑定錯誤、分析原因、修改與重測，可迅速修正錯誤。
  - 有效找出規格模糊、程式撰寫風格錯誤、不合規定程式、與無效程式段落等動態測試難以發現問題。
  - 審查與檢視透過小組討論可以有效交換成員觀念與想法，協助經驗傳承與訓練溝通。
- 動態測試
  - 須從系統非預期行為、或錯誤結果推斷錯誤程式碼.
  - 必須花許多時間診斷

審查項目

- 契約文件
- 計劃書(Project Plan, Configuration Management plan, ATP, STP, ITP, UTP)
- 軟體需求規格書(Software Requirement Specification)
- 軟體設計文件(Software Design Document)
- 原始程式碼(Source Code)
- 測試紀錄與錯誤報告(Test records and Fault reports)
- 建置紀錄(Configuration records)
- 維護手冊(Maintenance manual)
- 安裝程序手冊(Installation procedures)
- 使用者文件參考手冊(User documentation)

審查方法

- 檢視 (Inspection)
  - 最正規結構化(Formal structure)，程序包括計畫、準備、會議、更正-追蹤驗證、以及效果評估五階段。
  - 每個參與者都有固定的角色
  - 依據清楚的規格與檢查清單(Check list)，審查員事前單獨審查，之後再以會議方式共同討論找出問題
  - 指定修正解決人員，並產生報告。
- 團隊審查 (Team Review)、結構化逐步走查 (Structured Walkthrough)
  - 種輕量檢視(Inspection-lite)方法，比檢視方法不嚴謹。

審查方法

- 走查Walkthrough
  - 非正式(Informal)的審查方法，不需遵循既定程序
  - 由產品開發者主導，逐步說明產品如何執行典型應用案例
  - 審查者適時發問以追蹤產品設計，找出並記錄可能錯誤，
  - 指定錯誤處理負責人員，不需產生管理和度量(Metrics)報告。
- 程式走查 (Code Walkthrough, Code review)
  - 程式撰寫人員逐行講解程式設計的邏輯與格式
  - 與會人員充當電腦，實地走訪推敲程式運作的邏輯。
  - 測試人員提供重點測試案例，供分析執行流程與結果。
  - 提供常見錯誤一覽表，在會中並統計這些錯誤數據

審查方法

- 配對程式開發(Pair Programming)
  - 此即敏捷開發方法(Agile Methodology)的極限程式開發(Extreme Programming)
  - 每個開發工作都由兩位開發者一起進行。
- 桌上檢查(Desk Check)
  - 利用檢查表方式檢查。
  - 由程式設計人員進行單元測試前自主進行。
- 同儕桌上檢查(Peer Desk Check)和傳遞循環檢查(Pass Around)
  - 由同儕根據缺失檢查清單找出缺失，直接在產品上註記錯誤所在，或產生缺失報告。
- 特定問題審查(Ad hoc Review)。

Exercise Code Review

- 利用程式碼審查清單與程式碼寫作規範，審查程式碼。

審查階段 1

- 需求分析階段
  - 審查項目：部分或全部的需求規格
  - 重點：系統功能和性能
  - 目標：在初步設計前，找出需求描述不完整、模糊不精確、不具體、模擬兩可、不適當、不正確的需求描述
- 設計階段
  - 審查項目：部分或全部的軟體設計規格
  - 重點：系統架構、或元件結構功能和性能
  - 目標： 找出系統架構切割的錯誤、概念設計的錯誤、過度設計、或設計不符需求規格

審查階段 2

- 程式碼撰寫階段
  - 審查項目：虛擬程式碼、細部設計流程圖、程式碼正確性、複雜度
  - 重點：設計或實做的一小段
  - 目標：確認細部設計能正確滿足需求規格，和程式碼能正確滿足設計規格
  - 程序
    - 根據模組類型選定測試類別
    - 訂定Team/Personal標準 Check list
    - 根據 Check list複審程式碼，並提出建議報告
    - 統計錯誤報告清單，修改check list

審查階段 3

- 測試階段
  - 審查項目：測試計畫書、測試案例、測試程式碼。
  - 重點：各個測試層次，如單元測試、整合測試、接受測試
  - 目標：確保測試計畫書涵蓋軟體需求規範，找出測試案例的不足、測試案例的錯誤、測試程式碼的錯誤
- 測試計畫書審查
  - 重點：計畫書的完整性、一致性、彈性、清楚而不模糊
  - 不同測試階段，有不同的審查人員
  - 測試計畫人員、測試程式設計人員、品質保證人員、測試執行人員、市場代表、效能專家、終端使用者、領域專家、使用者手冊發展者

Review Roles

- 報告者–產生者或作者
  - 設定審查目標、準備資料、確認審查結果及後續處理動作
- 協調者(Coordinator) –協調會議、讓會議依議程順利進行
- 記錄者
  - 記錄會議所有建議、結論、和後續處理動作與負責人員，並在會後整理完記錄分給予會者
- 審查者–審查規格，找出問題
- 軟體品質保證人員(若需要)
  - 檢驗流程是否滿足品質管理標準程序.
- 使用者、客戶代表，(若需要)
  - 檢驗審查目標是否滿足使用者需求、或其他使用議題

Review Guideline

- 審查軟體開發程序或所開發的產品，不是審查開發者個人。
- 此為錯誤偵測而非問題解決會議。
- 牢記批評或給建議很容易，但是真正執行是困難的。
- 不要陷入辯論與反駁的口頭之爭。
- 須慎選人員，執行訓練，包含資訊技術與評論技巧。
- 尊重每位參與者發表意見的權利與內涵智慧。
- 依賴每位成員的責任感。
- 針對每項工作產品發展相對的檢查清單。
- 給審查員應有的審查工作時間與資源。
- 限制參與人員。
- 設定議程時間，並努力在預定議程時間內完成會議。
- 記錄會議中所有決議。

國立臺北科技大學資訊工程系郭忠義

審查流程–計畫與準備

- 計畫階段
  - 選擇審查團隊，平均3-4人，不超過8人，並分派角色。
  - 訓練審查團隊所需技巧。
  - 準備審查資料，包括規格書、工作產品(Work product)、審查checklist。
  - 設定品質標準
- 準備階段
  - 審查員審查資料，紀錄問題或錯誤。
  - 報告者(審查資料的作者)準備報告資料。

審查報告格式

- 審查報告又稱Observations
  - 報告撰寫日期、審查會議日期
  - 審查團隊姓名(Team name)
  - 審查產品名稱(Product Name)
  - 審查項目(Review items)
  - 審查目標(Objectives)
  - 處理項目(Action items)
  - 錯誤(Deficiencies)、遺漏(Omissions)、互相衝突(Contradictions)
  - 改善的建議
  - 針對每一個處理項目，必須記錄負責人、處理動作、問題嚴重程度(High / Medium / Low)、錯誤發生來源、目前解決狀態(Open/Closed)

審查流程–執行

- 審查會議約1小時，協調者根據議程主持會議討論，勿使會議成為『問題解決HOW』。
- 報告者(Reader) - 產品作者或其他人，逐句說明審查資料。
- 會中討論確認審查員找出的是否為問題或錯誤。
- 記錄者記錄Observations，包括錯誤、問題和審查意見。
- 會議決定
  - 問題處理方式、或問題處理計畫（單獨負責人、幾個人討論、解決期限、PLAN）
  - 問題種類、嚴重程度(severity)、與負責人員
  - 產品是否(1)合格、(2)小部分修改而接受、(3)需大修改而接受或重新審查、(4)取消重做。

審查流程–處理與追蹤

- 負責人員處理分配的問題(Action items)。
- 若審查結果為大部分修改，必須準備召開另一次會議
- 若審查結果為少部分修改而有條件接受，所有問題 (Action items) 須被審查者追蹤、判斷並更新問題(Action items)處理狀態
- 若有未處理或未解決的問題，必須提醒負責人員

審查流程–效果評估 (1)

- 審查可降低測試時間與增加測試品質，降低錯誤數量(預測最終軟體仍然存在的錯誤量)。
- 量化指標(Review Metrics)，統計大量審查資料
  - 審查項目的大小(Size)
  - 審查花費的時間
  - 錯誤發現的數量
  - 錯誤未發現的數量，但在之後的再次審查、或測試、或使用者使用時出現的錯誤(Defect)數量
  - 每個小時審查發現錯誤的數量
  - 每一頁或每一行(LOC)發現錯誤的數量

審查流程–效果評估 (2)

- 成本效用(Cost effectiveness)
  - 由審查者找出錯誤的成本，與此錯誤若留到後面修復，測試、與除錯所花的成本，兩者的比率。
- 錯誤密度(Defect Density)
  - 審查報告或錯誤報告中錯誤的數量，除以，預估程式的大小
- 錯誤偵測效率(Error defection efficiency)方程式
  - 審查記錄上，特定型態(reqs/design/code)的錯誤偵測數量/所有測試方法找出的錯誤數量
- Defect removal leverage (DRL)
  - 針對兩個不同階段，每小時錯誤移除的比率
  - 可比較出兩個不同階段，使用不同程序的相對效果