การทดสอบและการแก้จุดบกพร่อง

- เทคนิคการทดสอบ
- การออกแบบกรณีทดสอบ
- การแก้จุดบกพร่อง
- เครื่องมือช่วยการทดสอบและแก้จุดบกพร่อง

การทดสอบ

- การทดสอบโปรแกรมเป็นกระบวนการในการตรวจหาจุดบกพร่องของโปรแกรม
- ระดับการทดสอบจะสัมพันธ์กับขั้นตอนดำเนินการของกระบวนการพัฒนาโปรแกรม
- ส่วนสำคัญส่วนหนึ่งของการทดสอบ คือ กรณีและข้อมูลทดสอบ (Test cases and data)
- Test cases บอกถึงสถานการณ์ต่างๆ ที่โปรแกรมต้องตอบสนอง โดยต้องครอบคลุม ตั้งแต่ สถานะเริ่มต้น เหตุการณ์หรือภาวะการณ์ต่างๆ ที่มีผลต่อการดำเนินการของโปรแกรม ผลลัพธ์สุดท้ายที่คาดไว้
- ผลที่ได้ในขั้นตอนการวิเคราะห์ปัญหาหรือโจทย์ จะช่วยในการกำหนด Test cases ได้

ข้อผิดพลาด(Error)

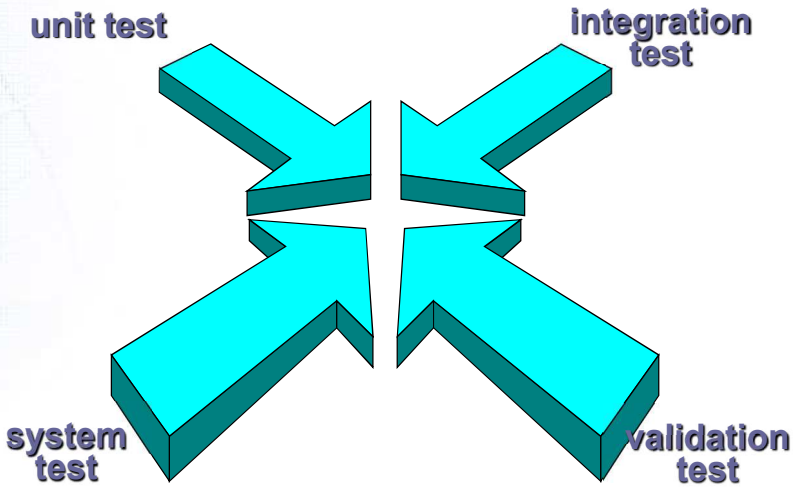
ข้อผิดพลาดที่เกิดขึ้นในการพัฒนาโปรแกรม

- Syntax error เป็นข้อผิดพลาดจากการเขียนผิดวากยสัมพันธ์ของภาษา
- Logical error เป็นข้อผิดพลาดที่เกิดขึ้นจากการทำงานของโปรแกรมซึ่งให้ผลไม่ถูกต้อง หรือไม่เป็นไปตามที่ต้องการ
- Runtime error เป็นข้อผิดพลาดที่เกิดขึ้นในขณะที่โปรแกรมถูกดำเนินการ โดยอาจจะเกิดเนื่องจากการทำงานที่ผิดเงื่อนไข หรือเป็นสภาวการณ์ที่ไม่คาดคิด เช่น การหารด้วยศูนย์

การทดสอบโปรแกรม

- การทดสอบโดยไม่ใช้คอมพิวเตอร์ (Manual Testing)
  - การทดสอบแบบตรวจการณ์ (Inspection)
    - โปรแกรมเมอร์ทำการทดสอบเอง โดยเปรียบเทียบชุดคำสั่งที่เขียน กับข้อผิดพลาดที่เคยปรากฏ
    - ทั้งนี้เพื่อ ไม่ให้เกิดข้อผิดพลาดแบบเดิมๆ
    - อาจจะไม่ทำให้ทราบว่าโปรแกรมทำงานถูกต้องหรือไม่
  - การทดสอบตามลำดับคำสั่งของโปรแกรม (Desk Checking)
    - ทดสอบโดยบุคคลอื่นที่ไม่ใช่ผู้เขียนโปรแกรม
    - โดยทดลองทำตามชุดคำสั่งที่เขียน เพื่อดูว่าโปรแกรมมีขั้นตอนการทำงานถูกต้องหรือไม่
    - ไม่เหมาะสำหรับโปรแกรมที่มีความซับซ้อน เพราะจะเสียเวลาในการทดสอบ
- การทดสอบแบบอัตโนมัติหรือโดยใช้คอมพิวเตอร์ (Automated Testing)
  - การทดสอบแบบหน่วย (Unit Testing)
  - การทดสอบแบบรวมหน่วยหรือเพิ่มหน่วย (Integration Testing)
  - การทดสอบระบบ (System Testing)

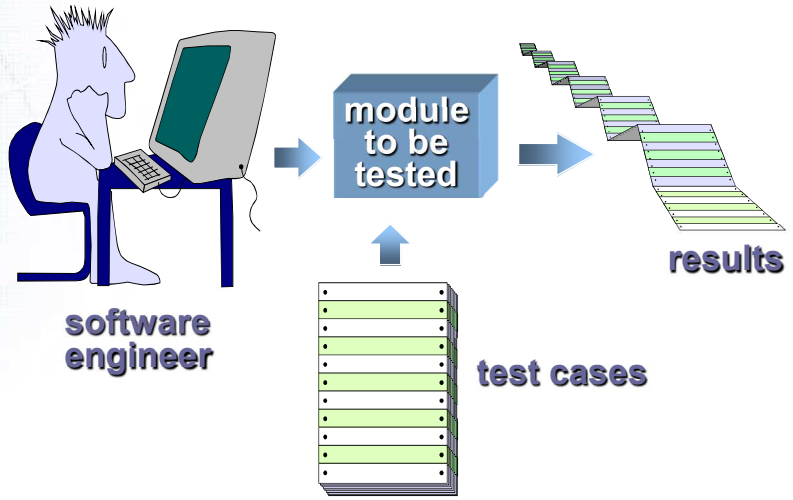
Testing Strategy



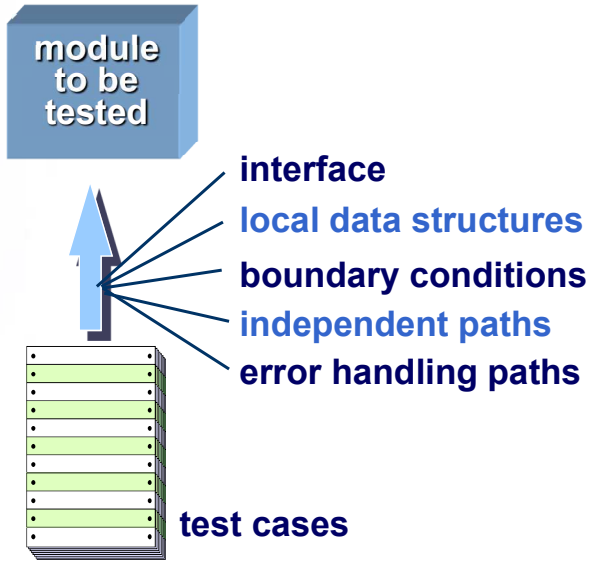
การทดสอบแบบหน่วย

- ทดสอบการทำงานทีละหน่วยหรือโมดูล
- ต้องการ โมดูลไดร์เวอร์ (Driver module) ในการทดสอบ
- โมดูลไดร์เวอร์ ทำหน้าที่
  - กำหนดค่าเริ่มต้นให้กับพารามิเตอร์ของโมดูลที่ถูกทดสอบ
  - เรียกโมดูลที่ต้องการทดสอบด้วยส่งผ่านค่าพารามิเตอร์ที่โมดูลนั้นต้องการ
  - รับค่ากลับ ที่เป็นผลจากโมดูลที่ถูกทดสอบ
- ในการทดสอบอาจจำเป็นต้องเขียนกลุ่มหรือชุดคำสั่งที่เขียนเพื่อแทนโมดูล เรียกกลุ่มคำสั่งนี้ว่า Stub

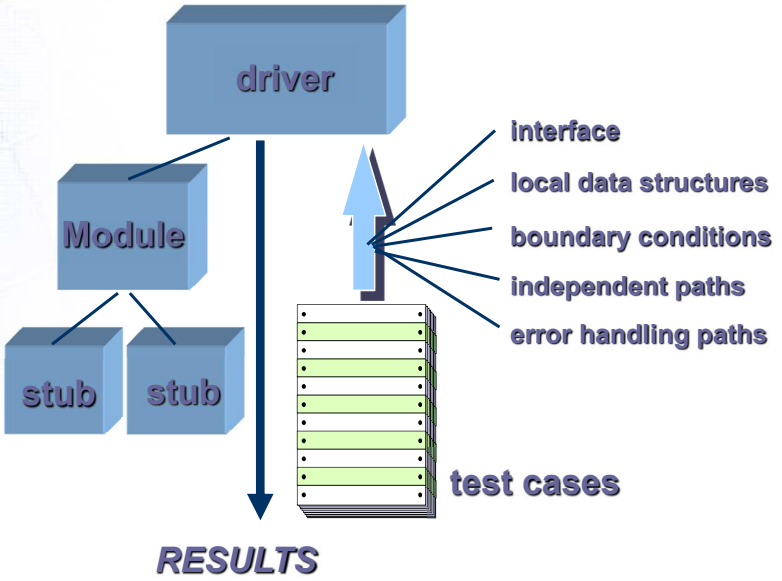
Unit Testing



Unit Testing



Unit Test Environment

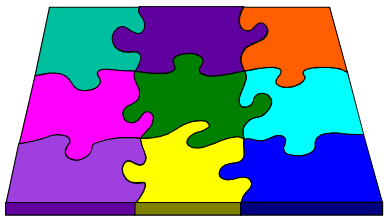


การทดสอบแบบรวมหน่วย

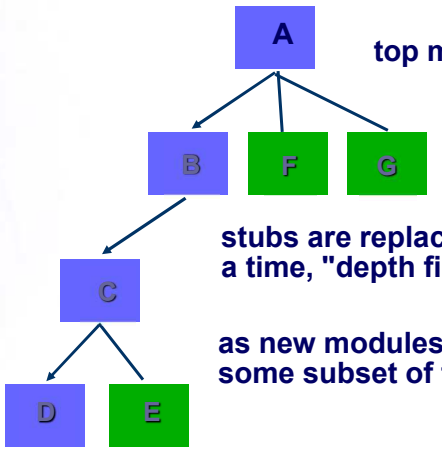
- ทดสอบการทำงานเมื่อมีการผูกรวมโมดูลเข้าด้วยกัน โดยการเพิ่มเข้าทีละโมดูล
- สามารถทดสอบการทำงานทั้งในสภาวการณ์ปกติ และกรณีที่โปรแกรมอาจจะมีปัญหาหรือเป็นกรณียกเว้น
- ข้อผิดพลาดที่อาจตรวจพบได้
  - ส่วนต่อประสานที่ไม่สอดคล้องกัน (Interface incompatibility)
  - การส่งผ่านค่าที่ไม่ถูกต้อง (Incorrect parameter values) เช่น ผิดชนิด หรือผิดสถานะ หรือผิดความหมาย เป็นต้น
  - Run-time exceptions
  - การตอบสนองหรือลักษณะการทำงานของโปรแกรมซึ่งไม่คาดว่าจะเกิดขึ้น (Unexpected state interactions)

การทดสอบแบบรวมหน่วย

- กลยุทธ์ในการรวมหน่วย มี 2 ทางเลือก
  - Big bang approach
  - Incremental construction strategy
    - Top-down approach
    - Bottom-up approach



Top Down Integration

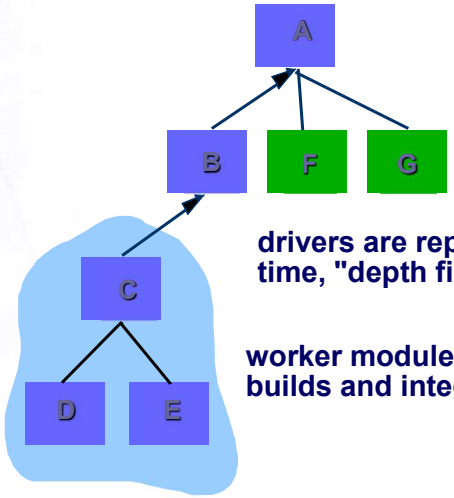


top module is tested with stub

stubs are replaced one at a time, "depth first"

as new modules are integrated, some subset of tests is re-run

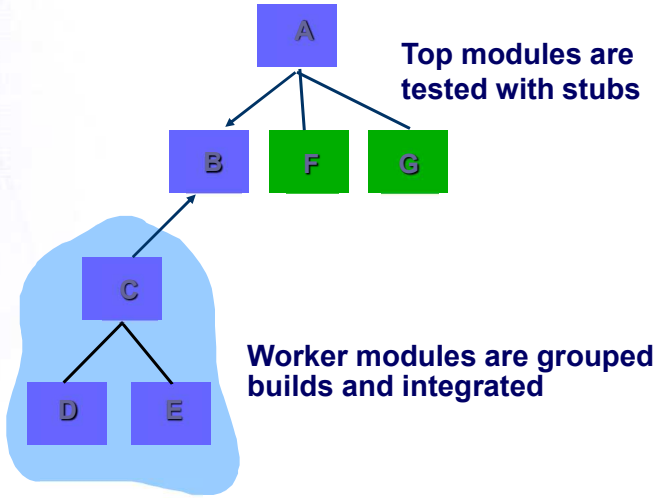
Bottom-Up Integration



drivers are replaced one at a time, "depth first"

worker modules are grouped into builds and integrated

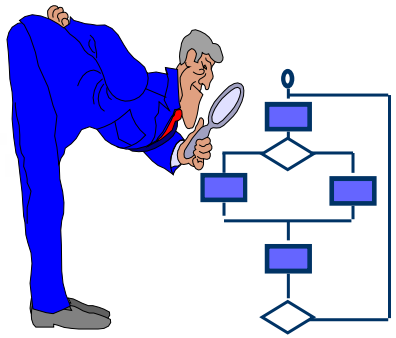
Sandwich Testing



Top modules are tested with stubs

Worker modules are grouped into builds and integrated

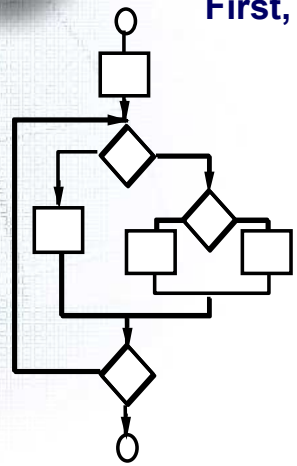
White-Box Testing



... our goal is to ensure that all statements and conditions have been executed at least once ...

Basis Path Testing

First, we compute the cyclomatic complexity :



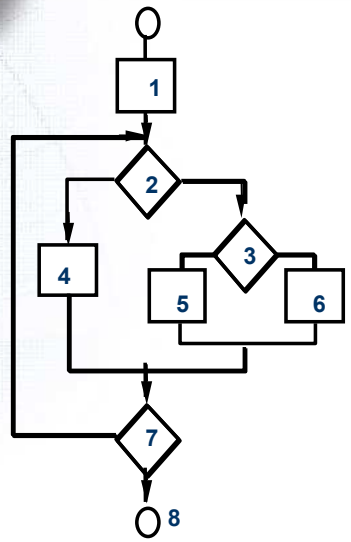
number of simple decisions + 1

or

number of enclosed areas + 1

In this case, V(G) = 4

Basis Path Testing



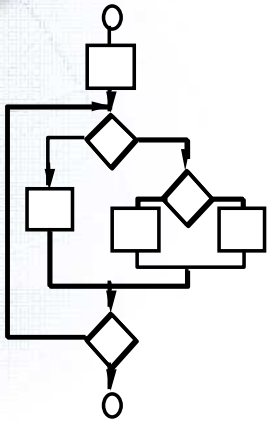
Next, we derive the independent paths:

Since V(G) = 4, there are four paths

Path 1: 1,2,3,6,7,8
Path 2: 1,2,3,5,7,8
Path 3: 1,2,4,7,8
Path 4: 1,2,4,7,2,4,...7,8

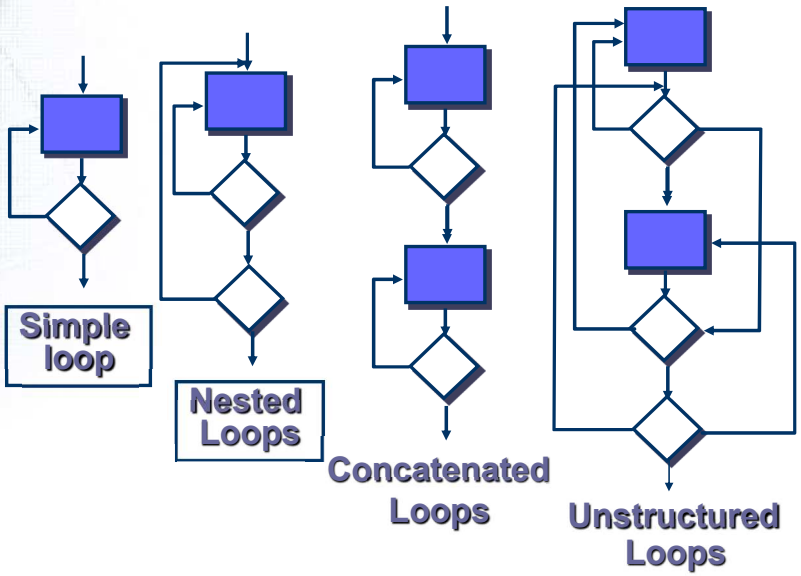
Finally, we derive test cases to exercise these paths.

Basis Path Testing Notes

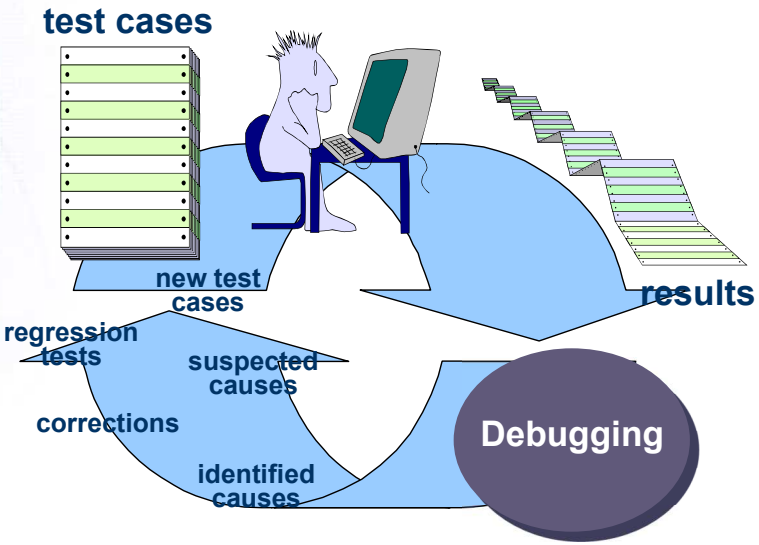


- you don't need a flow chart, but the picture will help when you trace program paths
- count each simple logical test, compound tests count as 2 or more
- basis path testing should be applied to critical modules

Loop Testing

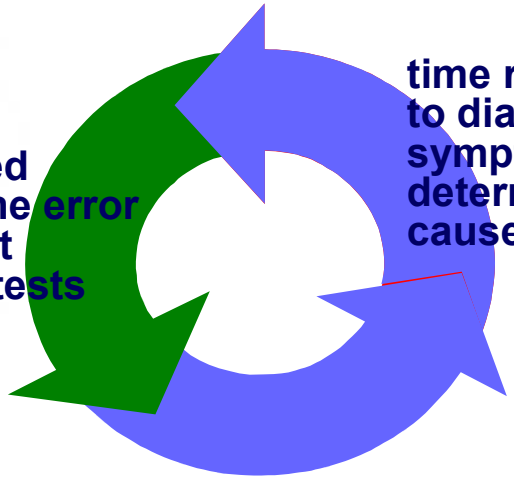


The Debugging Process



Debugging: A Diagnostic Process

Debugging Effort



time required to diagnose the symptom and determine the cause

time required to correct the error and conduct regression tests

Debugging: Final Thoughts

1. Don't run off half-cocked, think about the symptom you're seeing.
2. Use tools (e.g., dynamic debugger) to gain more insight.
3. If at an impasse, get help from someone else.