8. 확인과 검증

주요내용

- 품질이란 무엇인가?
- 검증과 확인이란 무엇인가?
- 동료 검토란 무엇인가?
- 테스팅이란 무엇인가?
- 블랙박스 테스팅 기법에는 무엇이 있는가?
- 화이트박스 테스팅 기법에는 무엇이 있는가?

목차

- 강의 내용
  - 검증과 확인
  - 품질 활동
  - 검증과 확인 기법

소프트웨어 개발과 품질

- 품질의 다양한 의미
  - 프로그램이 정상적으로 작동하는 것
  - 프로그램에 기대하는 막연한 완성도
  - 명시된 요구사항을 만족시키는 것
  - 고객이 의도한대로 요구사항을 올바르게 정의하는 것

검증과 확인 프로세스

검증과 확인 (1/2)

- 검증(Verification)
  - 올바른 제품을 생성하고 있는가?(Are we building the right product?) [Boehm]
  - 소프트웨어가 정확한 요구사항에 부합하여 구현되었음을 보장하는 활동
  - ‘요구사항 명세서에 맞게 올바른 방법으로 제품을 만들고 있음’ 을 보장

- 확인(Validation)
  - 제품이 올바르게 생성되고 있는가?(Are we building the product right?) [Boehm]
  - 소프트웨어가 고객이 의도한 요구사항에 따라 구현되었음을 보장하는 활동
  - ‘고객이 의도한 환경이나 사용 목적에 맞게 올바른 제품을 만들고 있음’ 을 보장

- 검증과 확인 작업은 실제로 구분하기 어려운 경우가 존재함
  - 결국, 소프트웨어의 품질을 보장하는 것

검증과 확인 (2/2)

❖ 검증과 확인 방법의 종류

– 정적(Static)인 방법

- 소프트웨어를 실행하지 않고 결함을 찾아내는 것
- 여러 참여자들이 모여 소프트웨어를 검토하여 결함을 찾아냄
- 소프트웨어 개발 중에 생성되는 모든 산출물들에 대해서 적용 가능
- 대표적인 방법
  - ➢ 검토(Review)
  - ➢ 인스펙션(Inspection)
  - ➢ 워크스루(Walk-through)

– 동적(Dynamic)인 방법

- 소프트웨어를 실행하여 결함을 찾아냄
- 발견된 결함은 디버깅 활동으로 확인하여 수정함
- 대표적인 방법
  - ➢ 테스팅

검증과 확인 기법

예제 시나리오

민서: 형 잠깐 시간 있어요? 이 프로그램에 버그를 못 잡겠어요.
잠깐 코드 좀 봐줄 수 있어요?
동석: 그래. 뭐가 문제야?
민서: 영어단어 프로그램인데 DB에 저장이 안돼요. 도대체 뭐가 문제인지 모르겠어요.
내 생각에는 DB연결 부분에 잘못된 것 같은데 30분이나 찾아봤지만 도저히
모르겠어요.
동석: 음.. 어디 보자. 그 부분엔 잘못된 것이 없어 보여.
여기를 봐봐. 쿼리를 만드는 스트링에서 따옴표 위치가 잘못되어 있잖아.
그러니까 제대로 된 쿼리가 안 만들어진 거야.
그것만 고치면 더 이상 오류는 안 날 것 같다.
민서: 아! 정말 그렇네. 정말 고마워요. 그 작은걸 하나 못 봤다니..

동료 검토(Peer review)란?

❖ 정의

- 개발 동료들이 검출된 결함의 개선을 위해 정의된 순서를 따르는 소프트웨어 작업 산출물을 검토하는 작업
  [SEI/CMU, “The Capability Maturity Model”, Addison-Wesley, 1994]
- 개발자가 자신의 동료들이 완료한 작업을 검토하는 것

❖ 목적

- 사용자 인터페이스 프로토타입, 요구 명세서, 아키텍처, 설계 및 기타 기술적 산출물의 품질 보증

동료 검토 절차 (1/4)

❖ 공지와 배포

- **산출물 작성자**
  - 자신의 산출물이 리뷰할 준비가 되었음을 검토자에게 통지 (예를 들어 프로젝트 계획서, 요구 분석, 사용자 인터페이스 프로토타입, 설계, 코드, 혹은 테스트 설계 등)
  - 공식 절차를 거쳐 해당 자료를 진행자(Moderator)에게 전달
- **진행자**
  - 산출물을 검토할 사람과 리뷰 회의에 참석할 사람을 결정
  - 검토를 위한 자료를 배포

❖ 준비

- **검토자는 이전에 가장 많이 발생했던 에러 체크리스트로 산출물을 검토**
- **검토 회의는 검토자들이 산출물에 대한 검토를 마친 후 개최**

❖ 검토

- **산출물의 작성자, 진행자, 검토자들이 모여 산출물을 검토**

동료 검토 절차 (2/4)

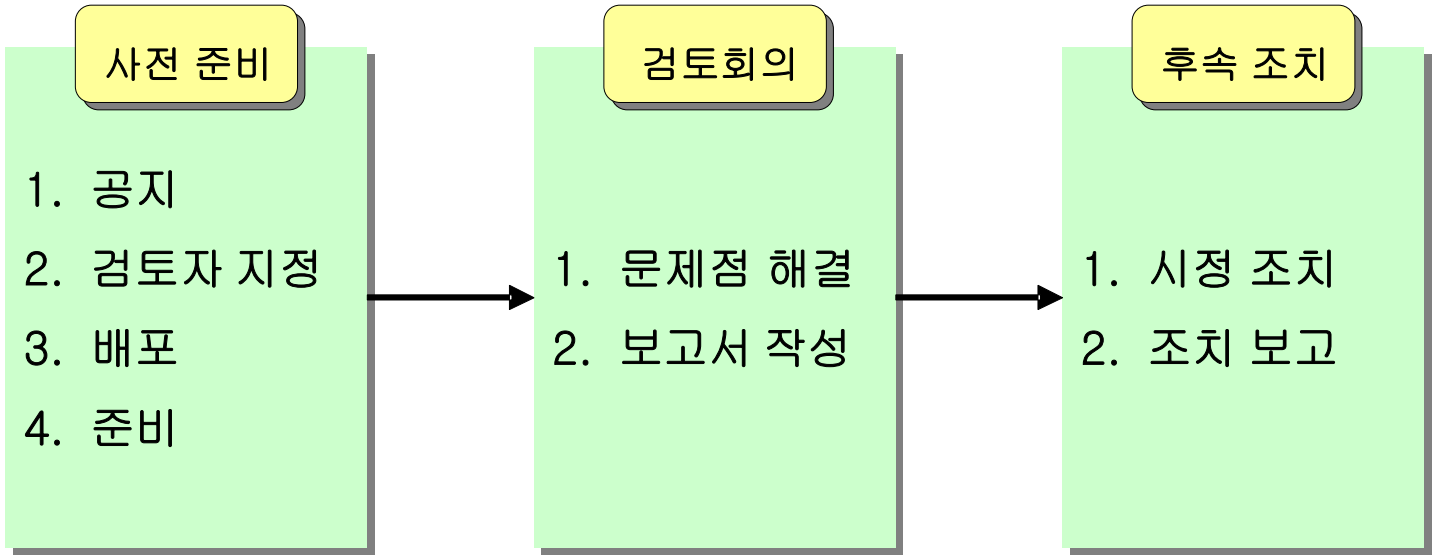
- **검토 보고서 작성**
  - 회의 후에 작성자와 진행자는 리뷰 회의의 결과 등을 기록으로 남김
  - 내용: 검토한 자료의 양, 발견된 결함의 종류와 개수, 회의에 걸린 시간, 산출물이 검토에 통과하였는지 여부

- **후속 조치**
  - 작성자 및 기타 관련자는 수정 작업을 끝내고, 변경된 내용을 검토한 후 해당 산출물이 공식적으로 검토 되었음을 선언

동료 검토 절차 (3/4)

❖ **동료 검토 프로세스**

- 체크리스트를 통해 검토를 진행하는 것이 좋음

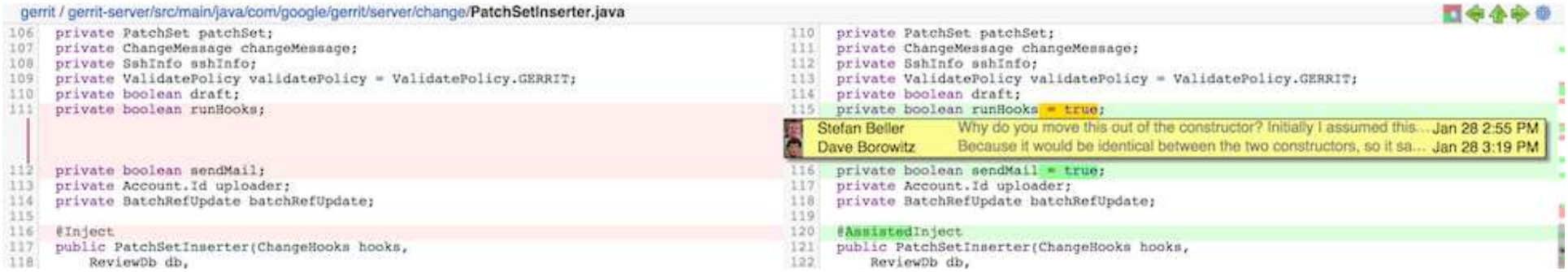


동료 검토 절차 (4/4)

❖ [예] 소스 코드에 대한 검토 체크리스트

번호	산출물 체크리스트
1	요구사항 명세서/정의서에 기술된 기능은 충분히 코드에서 실행되는가?
2	요구사항 명세서/정의서에 기술되지 않은 추가적인 기능이 코드에서 실행되지는 않는가?
3	메소드 리턴값은 적절하게 사용되는가?
4	모든 지역/전역변수(local and global variables)가 사용 전에 초기화되는가?
5	메소드가 정확히 호출되는가?

코드 리뷰 도구 활용



https://www.gerritcodereview.com/

소프트웨어 테스팅 (Software Testing)

테스팅(Testing)

❖ 정의

- 기존 조건 및 필요 조건(즉, 결함/에러/버그) 사이의 차이점을 발견하기 위하여 소프트웨어 항목을 분석하고, 분석된 항목의 특성을 평가하는 프로세스
  [IEEE-Std-829]
- 에러를 발견하려는 의도를 가지고 프로그램을 실행하는 프로세스
  [Myers]

결함을 지칭하는 다양한 용어

에러(Error), 오류

결함의 원인
사람(S/W 개발자, 분석가 등)에 의하여 생성된 실수가 주를 이룸

결함(Defect), 결점(Fault), 버그(Bug)

실패 또는 문제의 원인
제품에 포함된 결함

실패(Failure), 문제(Problem)

제품의 결함(Defect)이 있는 부분이 실행될 때 발생되는 현상

테스팅과 디버깅의 차이점

	테스팅(Testing)	디버깅(Debugging)
목적	알려지지 않은 에러의 발견	이미 알고 있는 에러의 수정
수행	시스템 내부 관련자, 테스팅 팀 등 외부의 제 3자	시스템 내부 관련자
주요 작업	에러 발견 (Error Detection)	에러의 정확한 위치 파악(Error Location) 에러의 타입 식별(Error Identification) 에러 수정(Error Correction)

http://www.redmine.org/issues/28807

테스트 케이스(Test Case)

❖ 의미

- 테스트의 목적에 맞게 테스트 조건, 입력값, 예상 출력값, 실제 테스트 결과를 기록하는 것[IEEE-Std-610]

❖ 목적

- 테스터가 테스팅을 체계적으로 할 수 있도록 함
- 개발자가 테스팅 결과를 통해 디버깅을 하는 기준이 됨

테스트 케이스의 예

테스트 케이스 ID: ST-0001					
목적	로그인 시 아이디와 패스워드의 대소문자를 구분하여 처리한다.				
테스트 조건	아이디/비번 : abcd / abcd 가 DB에 이미 입력되어 있음.				
테스터	한동석	테스트 일자	2006.10.01~2006.10.01		
단계	입력값	예상 출력값	실행 결과	조치사항	조치 후 시험결과
1	아이디: ABCD 패스워드: abcd	아이디 없음 경고	정상 로그인	디버깅 필요	아이디 없음 경고
2	아이디: abcd 패스워드: ABCD	패스워드 틀림 경고	패스워드 틀림 경고	-	-
3	아이디: abcd 패스워드: abcd	정상 로그인	정상 로그인	-	-

테스팅 종류

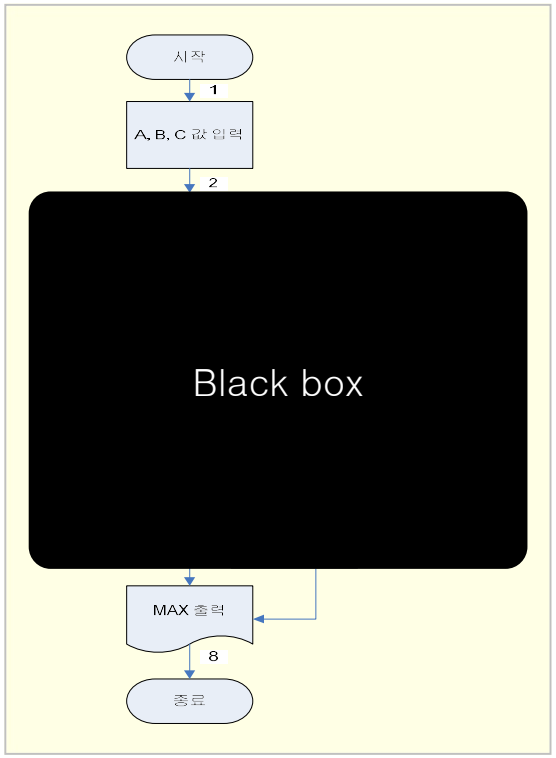
❖ 테스트 정보를 얻는 대상에 따른 분류

- **블랙박스 테스팅(Black-Box Testing)**
  - 요구사항 명세서(SRS)나 설계서로부터 테스트 케이스 추출
- **화이트박스 테스팅(White-Box Testing)**
  - 내부구조(소스 코드)를 기반으로 테스트 케이스 추출

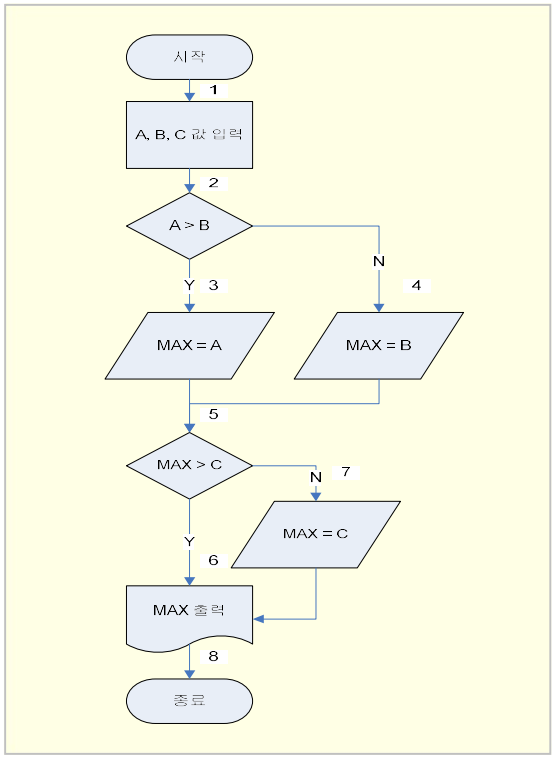
[예] 세 양수 중 큰 수 출력 프로그램 (1/3)

❖ 사용자 요구사항

- 서로 다른 세 양수 A, B, C를 입력하여 그 중 가장 큰 수를 출력하라.



(a) 블랙박스 테스팅



(b) 화이트박스 테스팅

[예] 세 양수 중 큰 수 출력 프로그램 (2/3)

❖ 블랙박스 테스트 케이스

ID	테스트 케이스
1	A가 가장 큰 정수일 경우…
2	B가 가장 큰 정수일 경우…
3	C가 가장 큰 정수일 경우…
4	A를 음수로 입력할 경우…
5	A와 B를 같은 정수로 입력할 경우…
6	B를 소수로 입력할 경우…
7	…

[예] 세 양수 중 큰 수 출력 프로그램 (3/3)

❖ 화이트박스 테스트 케이스

ID	테스트 케이스
1	(1->2->3->5->6->8)
2	(1->2->3->5->7->8)
3	(1->2->4->5->6->8)
4	(1->2->4->5->7->8)

블랙박스 테스팅(Black-Box Testing)

❖ 개요

- **요구사항 명세서나 설계서를 참조하면서 수행하는 테스팅**
  - 소스 코드 자체의 로직(Logic)에는 관심이 없고 입, 출력값에만 관심이 있다

- **방법**
  - 신택스 테스팅(Syntax Testing)
  - 동등분할(Equivalence Partitioning)
  - 경계값 분석(Boundary Value Analysis)
  - 의사결정 테이블(Decision Table)

신택스 테스팅(Syntax Testing)

❖ 개요

- 블랙박스 테스팅 기법 중 가장 단순한 방법
- 입력 데이터가 미리 정의된 데이터 유형에 적합한지를 검증하는 방법
- 입력 값을 적합(Valid)과 부적합(Invalid)으로 분류한 뒤, 예상되는 결과를 검증하는 기법

❖ 활용

- 회원 가입 폼과 같은 화면에서 사용자의 이름이나 아이디의 데이터 유형이 적합한지를 확인하는 경우

[예] 신택스 테스팅

입력 변수	적합 조건	부적합 조건
사용자 이름	6자리 이상 8자리 이하 알파벳	숫자, 특별 기호 등

No.	적합/부적합	입력 유형	입력 값	예상 결과
1	적합	7자리 알파벳	kildong	정상 처리
2	부적합	길이가 짧은 경우	kil	에러 메시지 - 길이 부적합
3	부적합	알파벳이 아닌 경우	길동386	에러 메시지 - 입력 유형 부적합

동등분할(Equivalence Partitioning)

❖ 개요

- 입력값이 범위가 정해져 있을 경우, 각 범위의 대표값을 이용하여 테스팅

❖ 장점

- 간단하고 이해하기 쉬움
- 이용자가 작성 가능
- 무작위 방법보다 체계적인 방법

[예] 동등분할 (1/2)

❖ 사용자 요구사항

- 100점이 만점이고 0 ~ 100점을 받을 수 있는 시험이 있다. 시험 점수를 입력하면, 점수에 따라 다음과 같이 A부터 F까지의 성적을 출력하라.

	성적
90점 이상 ~ 100점 이하	A
80점 이상 ~ 90점 미만	B
70점 이상 ~ 80점 미만	C
0점 이상 ~ 70점 미만	F

[예] 동등분할 (2/2)

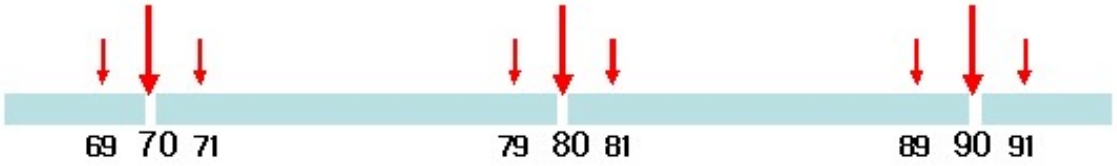
❖ 동등분할 테스트 케이스

	0≤ 점수 < 70	70 ≤ 점수 < 80	80 ≤ 점수 < 90	90 ≤ 점수 ≤ 100
테스트케이스	1	2	3	4
점수 범위	0점 이상 ~ 70점 이하	70점 이상 ~ 80점 미만	80점 이상 ~ 90점 미만	90점 이상 ~ 100점 이하
입력 값(점수)	50점	75점	85점	95점
예상 결과값	F	C	B	A
실제 결과값	F	C	B	A

경계값 분석(Boundary Value Analysis)

❖ 개요

- 입력 값의 주요 오류 대상인 경계값을 입력값으로 테스트 케이스를 작성하여 테스팅



❖ [예]

- 동등분할의 예제를 경계값 분석 방법을 이용하여 테스트 케이스를 추출한 경우

테스트케이스	1	2	3	4	5
입력 값(점수)	-1점	0점	99점	100점	101점
점수 범위	점수 범위 초과	정상	정상	정상	점수 범위 초과
예상 결과값	경고창	F	A	A	경고창
실제 결과값	경고창	F	A	A	경고창

의사결정 테이블(Decision Table)

❖ 개요

- 입/출력값이 True, False로 결정될 수 있는 경우 모든 경우의 수를 확인해볼 수 있는 방법

❖ 활용

- 입력, 출력 값이 Yes, No 로 결정 될 수 있는 경우
- 적은 수의 조건을 가진 입력값에 유용함

[예] 의사결정 테이블

❖ 사용자 요구사항

- 아이디와 비밀번호를 입력하여 둘 모두 유효하면 정상 로그인이 된다. 그러나 아이디가 유효하지 않을 경우 잘못된 아이디라는 경고창을, 아이디는 유효하나비밀번호가 유효하지 않으면 잘못된 비밀번호라는 경고창을 보여준다.

❖ 의사결정 테이블 테스트 케이스

테스트 조건		1	2	3	4
입력값	유효한 아이디	T	T	F	F
	비밀번호	T	F	T	F
예상 출력값	로그인 성공	T	F	F	F
	잘못된 아이디 경고창	F	F	T	T
	잘못된 비밀번호 경고창	F	T	F	F

화이트박스 테스팅(White-Box Testing)

❖ 개요

- **소스코드를 직접 참조하면서 수행하는 테스팅 기술**

- **방법**
  - 문장 커버리지(Statement Coverage)
  - 분기 커버리지(Branch Coverage)
  - 조건 커버리지(Condition Coverage)
  - 다중 조건 커버리지(Multiple Condition Coverage)
  - 기본경로 테스팅(Basic Path Testing)

문장 커버리지(Statement Coverage)

❖ 개요

- 프로그램을 구성하는 문장들이 최소한 한번은 실행될 수 있는 입력 값을 테스트 케이스로 선정함

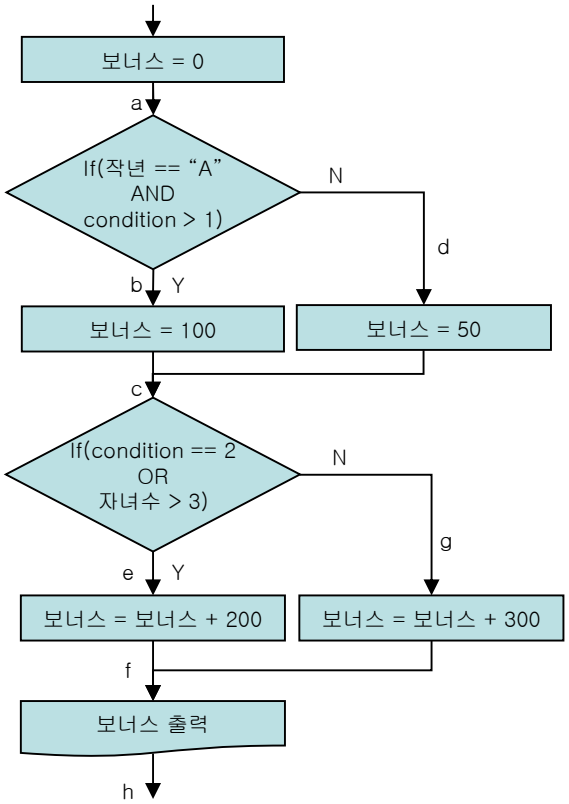
❖ [예]

- 테스팅 예제 순서도를 문장 커버리지를 적용하여 추출한 테스트 케이스

ID	테스트 케이스		
	입력값	경로	출력값
1	(A, 2, 2)	(a-b-c-e-f-h)	300
2	(A, 0, 4)	(a-d-c-e-f-h)	250
…	…	…	…

※ 입력 값은 (작년 유무, condition, 자녀 수) 이며, 출력 값은 보너스

A: 작년인 경우, B: 작년이 아닌 경우



< 테스팅 예제 순서도 >

분기 커버리지(Branch Coverage)

❖ 개요

- 프로그램에 있는 분기를 최소한 한번은 실행하게 하는 테스팅하는 방법

❖ [예]

- 테스팅 예제 순서도를 분기 커버리지를 적용하여 추출한 테스트 케이스

ID	테스트 케이스		
	입력값	경로	출력값
1	(A, 2, 2)	(a-b-c-e-f-h)	300
2	(B, 1, 2)	(a-d-c-g-h)	0
…	…	…	…

조건 커버리지(Condition Coverage)

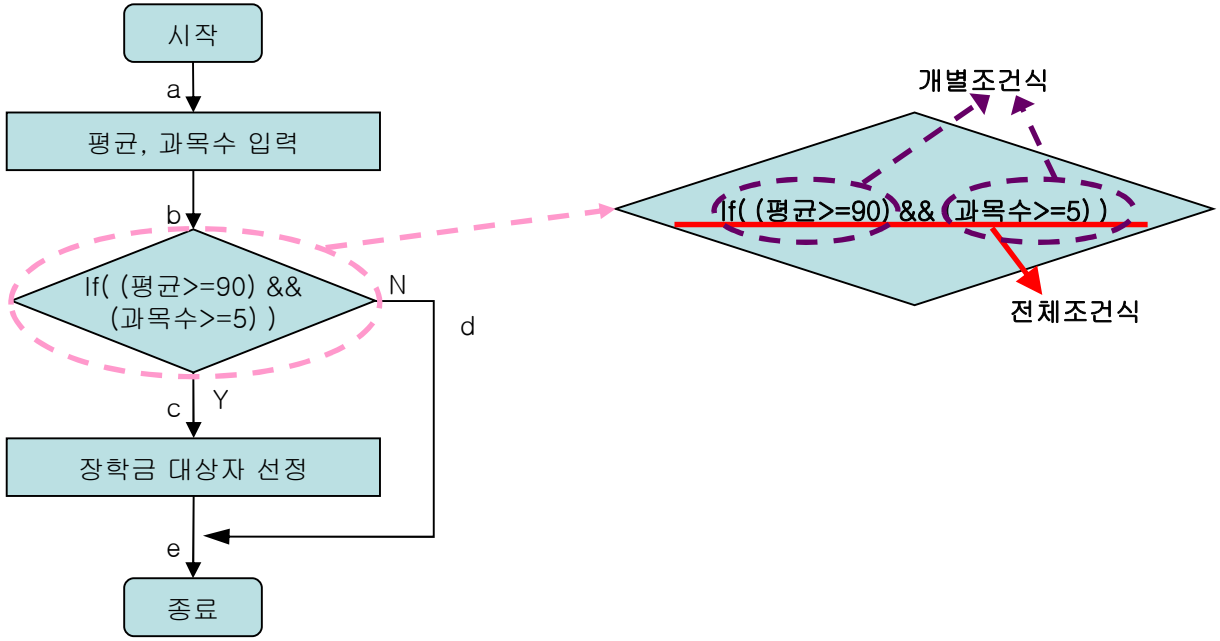
❖ 개요

- &&, || 등의 조건을 가진 분기문이 전체 조건식의 결과와 관계없이 &&나 || 전후의 각 개별 조건식이 참 한 번, 거짓 한 번을 갖도록 테스트 케이스를 만드는 방법

[예] 조건 커버리지 (1/2)

❖ 사용자 요구사항

- 학생의 평균과 과목수를 받아서, 장학금 대상자를 선정하라. 장학금 대상자는 평균이 90점 이상이고, 과목수가 5과목 이상인 학생으로 한다.



[예] 조건 커버리지 (2/2)

❖ 조건 커버리지 테스트 케이스

ID	테스트 케이스		
	입력값	경로	출력값
1	(95, 4)	(a-b-d-e)	대상자 아님
2	(72, 7)	(a-b-d-e)	대상자 아님
…	…	…	…

※ 입력값은 (평균, 과목수)이며, 출력값은 대상자 선정 여부이다.

❖ 조건 커버리지 테스트 케이스 진리표

평균	과목수	전체조건식
95 이면 참	4 이면 거짓	거짓
72 이면 거짓	7 이면 참	거짓
…	…	…

기본경로 테스팅(Basic Path Testing)

❖ 개요

- Tom McCabe에 의해 개발된 기법
- 프로그램의 제어구조(Control Structure)를 플로우 그래프(Flow Graph)로 표현하고, 순환복잡도(Cyclomatic Complexity)를 통해 독립적인 경로의 수를 찾아 테스트 케이스를 추출하는 기법

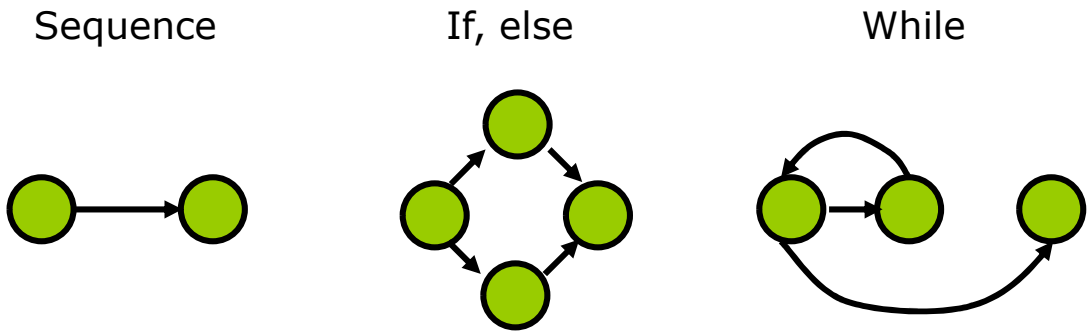
❖ 테스트 케이스 추출 단계

1. 테스팅 할 대상의 플로우 그래프를 그린다.
2. 순환복잡도를 계산한다.
3. 독립적인 경로들을 정의한다.
4. 정의된 각 경로의 테스트 케이스를 작성한다.

1. 테스팅 할 대상의 플로우 그래프를 그린다.

❖ 개요

- **입력에서 출력에 이르기까지의 프로그램 내부구조를 플로우 그래프 (Flow Graph)로 표현**
  - 플로우 그래프
    - ➢ 프로그램 내부구조의 제어흐름(Control Flow)을 그래프로 표현하여 구조를 파악하고 시험 경로(Path)를 추출하기 위해 표현
  - 플로우 그래프 표기법



2. 순환복잡도를 계산한다.

❖ 개요

- **순환복잡도(Cyclomatic Complexity)를 통해 전체 프로그램 내부구조를 시험 할 수 있는 독립적인 경로의 수를 계산**
  - 순환복잡도프로그램의 논리적인 복잡도를 정량적으로 측정하기 위해 제공되는 매트릭(Matric)
  - 순환복잡도 공식

• CC = R의 수
• CC = E - N + 2
• CC = P + 1

CC(Cyclomatic Complexity): 순환복잡도
R(Region): 노드와 가장자리 노드로 둘러싸인 영역과 그래프 밖 영역의 수
E(Edge): 화살표의 수
N(Node): 노드의 수
P(Predicate): 분기 노드의 수

3. 독립적인 경로들을 정의한다.

❖ 개요

- 순환복잡도를 통해 계산된 횟수를 기반으로 독립적인 경로들을 정의

4. 정의된 각 경로의 테스트 케이스를 작성한다.

❖ 개요

- 정의된 각 경로를 시험할 테스트 케이스를 작성

[예] 기본 경로 테스팅 (1/5)

❖ 요구사항

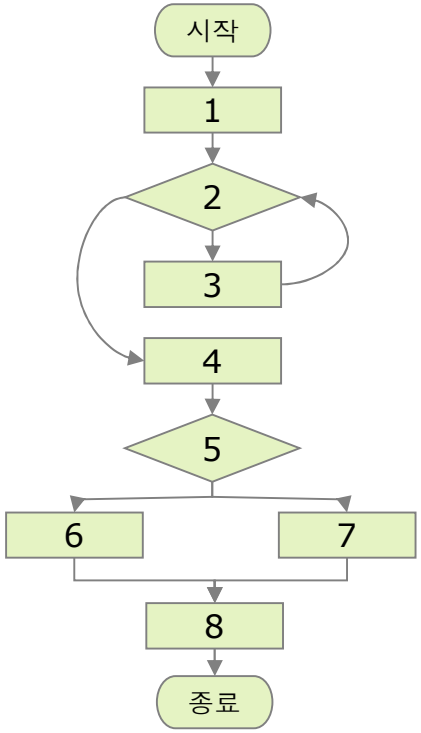
- 국어, 영어, 수학 점수를 입력 받아, 평균점수가 70점 이상이면 'PASS'를, 미만이면 ' FAIL'을 출력한다

```
int main(int argc, char *argv[])
{
int[] testPoint = input[]

while(testpoint.count() =! 3)
{
   int inputPoint;
   printf("점수를 입력해주세요");
   scnaf("%d", &inputPoint);
   testPoint.add() = inputPoint;
}

int avPoint = (testPoint[0] + testPoint[1] + testPoint[2]) / 3;

if(avPoint >= 70)
{
   printf("PASS");
}
else
{
   printf("FAIL");
}
printf("출력이 종료되었습니다")
```

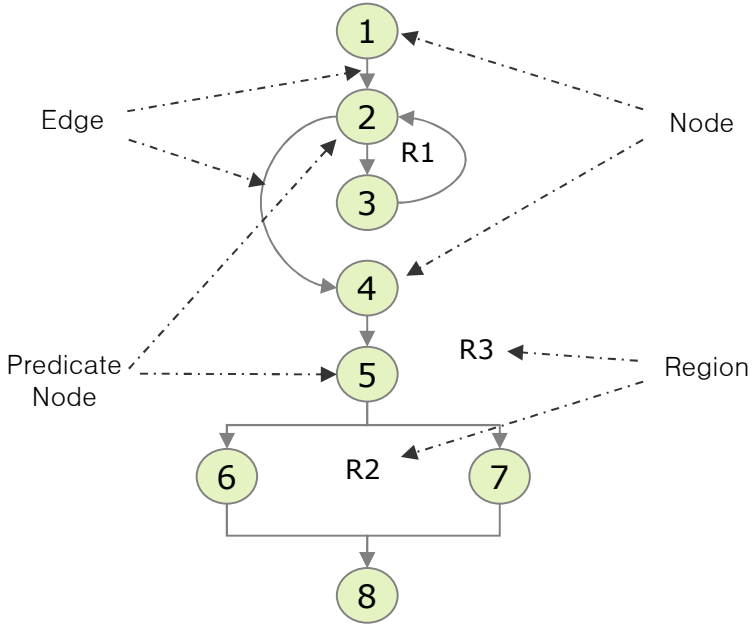
[예] 기본 경로 테스팅 (2/5)

1. 테스팅 할 대상의 플로우 그래프를 그린다.

- 3과목 평균점수의 PASS여부 예제의 순서도와 플로우 그래프



(A) 순서도(Flow chart)



(B) 플로우 그래프(Flow graph)

[예] 기본 경로 테스팅 (3/5)

2. 순환복잡도를 계산한다.

- 순환복잡도 공식에 따라,
  - V(G) = 3
  - V(G) = 9 - 8 + 2 = 3
  - V(G) = 2 + 1 = 3

- 계산 결과 순환복잡도는 3이므로, 최소한 3번 이상의 독립적인 경로를 시험하면 전체 내부구조를 포함할 수 있음

[예] 기본 경로 테스팅 (4/5)

3. 독립적인 경로들을 정의한다.

- **순환복잡도를 통해 계산된 횟수를 기반으로 독립적인 경로들을 정의하면,**
  - 경로 1: 1 - 2 - 3 - 2 - 4 - 5 - 6 - 8
  - 경로 2: 1 - 2 - 3 - 2 - 4 - 5 - 7 - 8
  - 경로 3: 1 - 2 - 4 - 5 - 6 - 8

[예] 기본 경로 테스팅 (5/5)

4. 정의된 각 경로의 테스트 케이스를 작성한다.

- **정의된 각 경로를 시험할 테스트 케이스를 작성하면,**
  - 3과목 평균점수의 PASS여부 예제의 테스트 케이스

ID	테스트 케이스		
	경로	입력값	예상 출력값
1	경로 1	선 입력{70, 75}, 후 입력{80}	PASS
2	경로 2	선 입력{60, 65}, 후 입력{50}	FAIL
3	경로 3	선 입력{70, 90, 80}, 후 입력{없음}	PASS

※ 입력값은 (국어, 영어, 수학)으로 선,후의 입력된 조건이 있으며, 출력값은 PASS 여부이다.

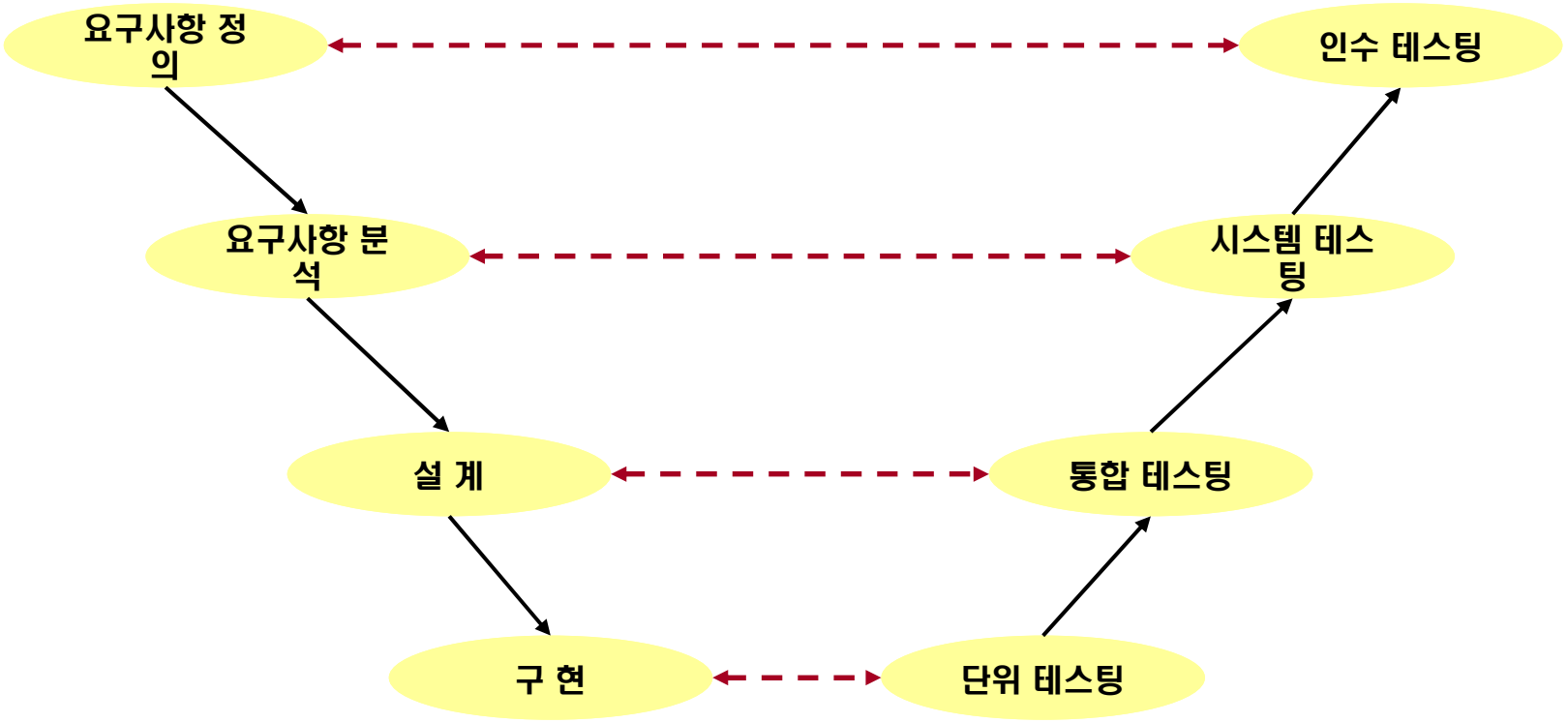
화이트박스 테스팅의 특징

- 테스트의 목적과 조건에 맞게 적절한 방법 선택
- 각 테스팅 방법에 따라 복잡도, 소요되는 시간(비용)이 다름

테스팅 단계

소프트웨어 개발 단계와 테스팅

❖ 소프트웨어 개발 단계마다 생산되는 산출물을 이용하여 테스팅 수행



테스팅 단계

❖ 단위 테스팅(Unit Testing)

- **개요**
  - 구현 단계에서 각 모듈이 완성되었을 경우 개별적인 모듈을 테스팅
  - 테스팅의 주체는 해당 모듈의 개발자
  - 화이트박스/블랙박스 테스팅 모두 가능
- **테스트 할 모듈을 단독적으로 실행할 수 있는 환경 필요**
  - 스텁(Stub)
    - ➢ 테스팅 대상 모듈에서 호출하는 모듈
  - 테스트 드라이버(Test Driver)
    - ➢ 테스팅 대상 모듈을 호출하는 환경

테스팅 단계

❖ 통합 테스팅(Integrating Testing)

- **개요**
  - 모듈을 통합한 단계에서 수행되는 테스팅
  - 모듈간의 상호작용을 검사하는 테스팅
- **모듈 통합 방법에 따른 테스팅 기법 종류**
  - 빅뱅(Big Bang) 기법
    - ➢ 모듈을 한꺼번에 통합하여 테스팅을 하는 방법
    - ➢ 오류가 발생하였을 경우 어느 부분에서 오류가 났는지 찾기가 어려움
  - 하향식(Top-Down) 기법
    - ➢ 가장 상위 모듈부터 하위 모듈로 점진적으로 통합하는 방법
    - ➢ 상위 모듈 테스팅 시, 하위 모듈에 대한 스텁이 필요
  - 상향식(Bottom-Up) 기법
    - ➢ 하위 모듈부터 테스팅 하고 상위 모듈로 점진적으로 통합하는 방법
    - ➢ 하위 모듈 테스팅 시, 상위 모듈에 대한 테스트 드라이버가 필요

테스팅 단계

❖ 시스템 테스팅(System Testing)

- **개요**
  - 모듈이 모두 통합된 후, 사용자의 요구사항이 만족되었는지 검사하는 테스팅
  - 고객에게 시스템을 전달하기 전, 시스템을 개발한 조직이 주체가 되는 마지막 테스팅
- **테스팅 대상**
  - 요구사항 명세서를 기초로 하여 사용자의 기능 요구사항
  - 보안, 성능, 신뢰성 등의 비기능 요구사항

테스팅 단계

❖ 인수 테스팅(Acceptance Testing)

- **개요**
  - 시스템이 사용자에게 인수되기 전, 사용자에 의해 실시되는 테스팅
  - 실제 사용자가 운영하는 환경에서 실시
  - 인수 테스팅을 통과해야만 시스템이 정상적으로 사용자에게 인수되고 프로젝트는 종료됨

연습문제

1. 모듈 안의 작동을 자세히 관찰할 수 있으며, 프로그램 원시 코드의 논리적인 구조를 커버하도록 테스트 케이스를 설계하는 프로그램 테스트 방법은 무엇인가?

2. 블랙박스 테스팅은 무엇인가?

3. 소프트웨어의 테스팅 중 화이트 박스 테스팅의 과정은 무엇이 있는가?

4. 테스팅 단계의 순서를 나열하라.

5. 디버깅(Debugging)이란 무엇인가?

팀 프로젝트

12, 13주차

이번 주 할일

- 각 팀은 개발한 소프트웨어 시스템의 확인과 검증을 한다
  - 단위 테스트와 통합 테스트 실시
  - 각 테스트 결과 보고서 작성

- 평가 기준 (5점 만점)

- 결과
  - 3.5점 이상이면 통과 함

다음 주 제출 문서

- 단위 테스트 및 통합 테스트의 테스트 결과 보고서 제출