

9주 2강

동적 테스트, 소프트웨어 개발 단계에 따른 테스트



이번 주차에는...

실전 적용 시뮬레이션

- 동적 테스트
- 소프트웨어 개발 단계에 따른 테스트

1. 명세 기반 테스트

■ 명세 기반 테스트(블랙박스 테스트)



그림 8-14 블랙박스 테스트 비유: 청진기로 환자 진찰하기

- 입력 값에 대한 예상출력 값을 정해놓고 그대로 결과가 나오는지 체크
- 프로그램 내부의 구조나 알고리즘을 보지 않고, 요구 분석 명세서나 설계 사양서에서 테스트 케이스를 추출하여 테스트
- 기능을 어떻게 수행하는가 보다는 사용자가 원하는 기능을 수행하는가 테스트

2. 동적 테스트

- 동적 테스트 dynamic test
 - 테스트 데이터를 이용해 실제 프로그램을 실행함으로써 오류를 찾는다.
- 정보를 얻는 문서 종류에 따른 분류
 - 명세 기반 테스트
 - 구현 기반 테스트

3. 선택스 기법

■ 선택스 syntax 기법

- 문법에 기반을 둔 테스트
- 문법을 정해놓고 적합/부적합 입력 값에 따른 예상 결과가 제대로 나오는지 테스트

표 8-3 ID와 비밀번호의 적합 기준

	적합	부적합
ID	알파벳과 숫자(4자~8자)	특수 기호 사용, 4자 미만, 9자 이상
비밀번호	알파벳 · 숫자 · 특수문자로 구성, 7글자 이상, 특수문자는 반드시 포함.	7자 미만, 특수문자 미사용

표 8-4 적합/부적합 입력 값에 대한 예상 처리 결과

번호	적합/부적합	입력 값(ID/비번)	예상 처리 결과
1	적합	sogong/abcd#78	정상 등록
2	부적합	so*gong/abcd#78	에러 메시지: 등록 불가(ID-특수문자 사용)
3	부적합	sogong/abcdefg	에러 메시지: 등록 불가(비밀번호-특수문자 미사용)
4	부적합	gong/abcd	에러 메시지: 등록 불가(ID, 비밀번호-길이 부적합)

4. 동등 분할 기법

■ 동등 분할^{equivalence partitioning} 기법

- 각 영역에 해당하는 입력 값을 넣고 예상되는 출력 값이 나오는지 실제 값과 비교
(즉) 입력 값에 대한 예상 결과 값을 미리 정해 놓고, 각 영역에서 임의의 값을 하나 정해 입력 값으로 사용하여 실제 결과가 예상 값과 같은지 확인
- 장점: 단순하고 이해하기 쉬우며 사용자가 작성 가능

표 8-5 평가 점수에 따른 상여금

평가 점수	상여금
90~100점	600만 원
80~89점	400만 원
70~79점	200만 원
0~69점	0원

표 8-6 동등 분할 기법의 예

테스트 케이스	1	2	3	4
점수 범위	0~69점	70~79점	80~89점	90~100점
입력 값	60점	73점	86점	94점
예상 결과 값	0원	200만 원	400만 원	600만 원
실제 결과 값	0원	200만 원	400만 원	600만 원

5. 경계 값 분석 기법

- 경계 값 분석(boundary value analysis)기법
 - 경계에 있는 값을 테스트 데이터로 생성하여 테스트하는 방법
(즉) 경계 값과 경계 이전 값, 경계 이후 값을 가지고 테스트

표 8-7 경계 값 분석 기법의 테스트 예

테스트 케이스	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
입력 값	-1	0	1	69	70	71	79	80	81	89	90	91	99	100	101
예상 결과	오류	0	0	0	200	200	200	400	400	400	600	600	600	600	오류
실제 결과	오류	0	0	0	200	200	200	400	400	400	600	600	600	600	오류

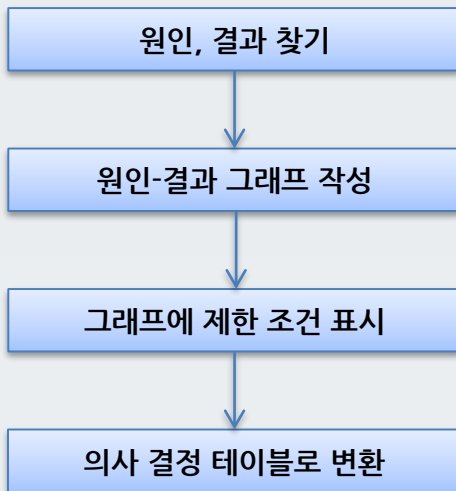
6. 원인-결과 그래프 기법(1)

- 동등 분할/경계 값 분석 기법의 단점
 - 입력 환경의 복잡성을 완전하게 고려하지 못함



단점 극복

원인-결과 그래프(cause-effect graph)기법



7. 원인-결과 그래프 기법(2)

- 프로그램을 적합한 크기로 분할한다
 - 규모가 큰 프로그램은 원인-결과 그래프를 작성할 만한 크기로 분할한다.
- 원인과 결과를 찾는다
 - 명세서에서 원인과 결과를 찾아 일련번호와 같은 식별자를 각각 부여한다. 여기서 원인은 하나의 입력 조건이고, 결과는 출력 조건이다.
- 원인-결과 그래프를 작성한다
 - 프로그램 명세가 의미하는 내용을 분석하여, 이에 알맞은 원인과 결과를 연결하는 논리 그래프를 작성한다





8. 원인-결과 그래프 기법(3)

표 8-8 원인-결과 그래프에서 사용되는 기호

기호	표기법	사용 예	설명
동치(Equal, -)	$A \sim B$	if A then B	A=1이면 B=1, A=0이면 B=0
부정(NOT, ~)	$A \sim B$	if not A then B	A=1이면 B=0, A=0이면 B=1
합(OR, ∨)	$A \vee B$	if A or B then C	(A, B, C)에서 (1, 1)=1, (1, 0)=1, (0, 1)=1, (0, 0)=0
곱(AND, ∧)	$A \wedge B$	if A and B then C	(A, B, C)에서 (1, 1)=1, (1, 0)=0, (0, 1)=0, (0, 0)=0

■ 그래프에 제한 조건을 표시한다.

표 8-9 원인-결과 그래프의 제한 조건 기호

명칭	기호	설명	
배타적 ^{Exclusive} 관계	E 	두 개가 동시에 존재할 수 없다.	A=1이면 B=0, A=0이면 B=1 (A=B=0은 가능, A=B=1은 불가능)
포함 ^{Inclusive} 관계	I 	적어도 둘 중 한쪽은 성립한다.	A=1일 때 B=1, B=0 B=1일 때 A=1, A=0
선택 ^{Only one} 관계	O 	항상 한쪽만 성립한다.	A=1이며 B=0, 또는 A=0이며 B=1 (A=B=0은 불가능)
필요 ^{Require} 관계	R 	한쪽이 성립하면 다른 쪽도 성립한다(B가 성립하기 위해서는 A가 반드시 필요하다).	A=1이면 B=1 또는 B=0이고 A=1이면 B=0 (B=1은 불가능)
강요 ^{Mask} 관계	M 	한쪽이 성립하면 다른 쪽은 성립하지 않는다.	A=1이면 B=0

9. 원인-결과 그래프 기법(4)

- 위사결정 테이블로 변환한다.

표 8-10 입력 항목에 대한 테스트 케이스

테스트 케이스		1	2	3	4	5	6	7	8
입력 항목									
총점	90 이상	T	T	F	F	F	F	F	F
	80 이상	F	F	T	T	F	F	F	F
	70 이상	F	F	F	F	T	T	F	F
	70 미만	F	F	F	F	F	F	T	T
리포트	제출	T	F	T	F	T	F	T	F

- 1 : 90점 이상(T), 리포트 제출(T) → A1
- 2 : 90점 이상(T), 리포트 미제출(F) → A0
- 3 : 80점 이상(T), 리포트 제출(T) → B1
- 4 : 80점 이상(T), 리포트 미제출(F) → B0
- 5 : 70점 이상(T), 리포트 제출(T) → C1
- 6 : 70점 이상(T), 리포트 미제출(F) → C0
- 7 : 70점 미만(T), 리포트 제출(T) → D
- 8 : 70점 미만(F), 리포트 미제출(F) → F

10. 원인-결과 그래프 기법(5)

성적 처리에 대한 의사결정 테이블

표 8-11 성적 처리에 대한 의사결정 테이블

입력 항목		테스트 케이스							
		1	2	3	4	5	6	7	8
총점	90 이상	T	T	F	F	F	F	F	F
	80 이상	F	F	T	T	F	F	F	F
	70 이상	F	F	F	F	T	T	F	F
	70 미만	F	F	F	F	F	F	T	T
리포트	제출	T	F	T	F	T	F	T	F
결과 값	A ⁺	T	F	F	F	F	F	F	F
	A ⁰	F	T	F	F	F	F	F	F
	B ⁺	F	F	T	F	F	F	F	F
	B ⁰	F	F	F	T	F	F	F	F
	C ⁺	F	F	F	F	T	F	F	F
	C ⁰	F	F	F	F	F	T	F	F
	D	F	F	F	F	F	F	T	F
	F	F	F	F	F	F	F	F	T

11. 원인-결과 그래프 기법(6)

- 테스트 케이스를 작성한다
 - 작성된 의사결정 테이블을 기반으로 테스트 케이스를 도출한다.
(예) 테스트 케이스3번째의 테스트 데이터(총점 86점, 리포트 제출)를 생성했다면 결과 값이 B1가 되어야 한다

12. 원인-결과 그래프 예제

■ [문제]

- 첫 번째 열이 P 또는 S로 시작하고, 두 번째 열이 #이면 '출입 가능'을 출력한다.
- 첫 번째 열이 P 또는 S로 시작하지 않으면 '출입 금지'를 출력한다.
- 첫 번째 열이 P 또는 S로 시작하고, 두 번째 열이 #이 아니면 비'밀번호 오류'를 출력한다.
- P는 교수(Professor), S는 학생(Student)을 의미한다.

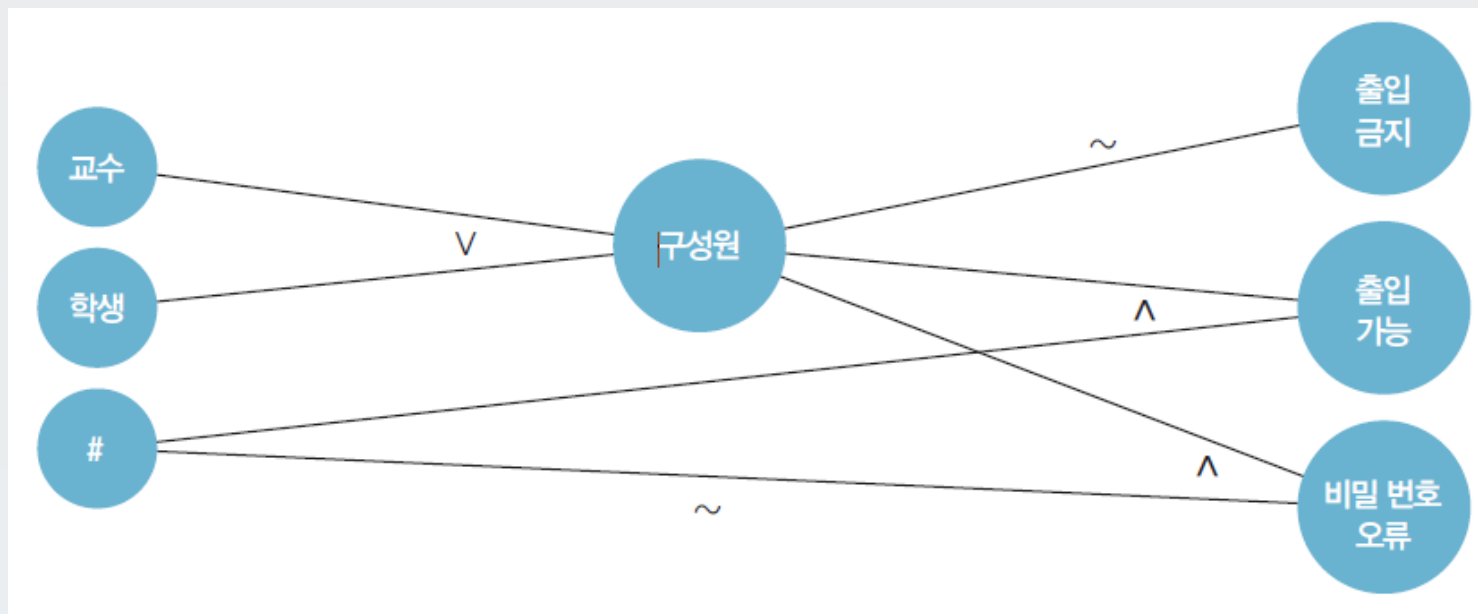
■ [풀이]

- 원인과 결과를 찾는다.

원인	결과
원인 1 : 첫 번째 열이 P	결과 1 : '출입 가능' 출력
원인 2 : 첫 번째 열이 S	결과 2 : '출입 금지' 출력
원인 3 : 두 번째 열이 #	결과 3 : '비밀번호 오류' 출력

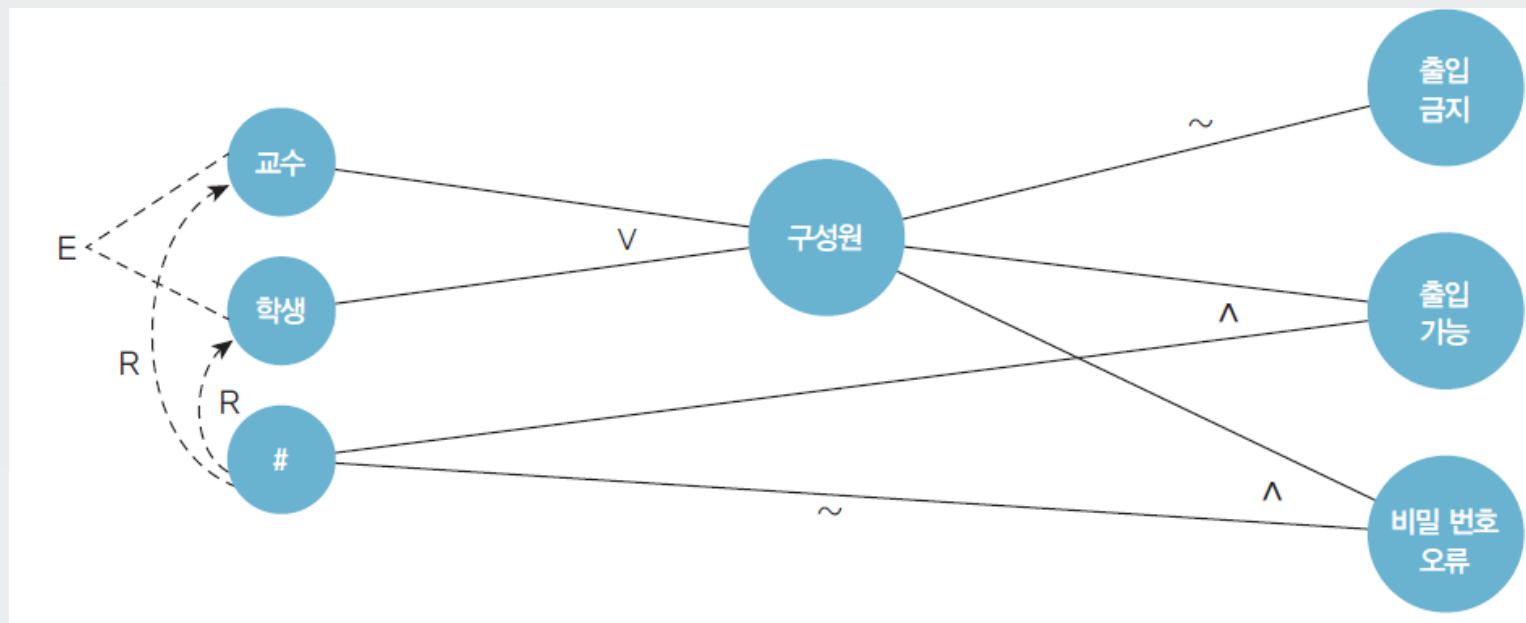
13. 원인-결과 그래프 예제

- [풀이] 원인-결과 그래프를 작성한다



14. 원인-결과 그래프 예제

- [풀이] 원인-결과 그래프의 제한 조건을 표시한다.



15. 원인-결과 그래프 예제

- [풀이] 의사 결정 테이블로 변환한다.

테스트 케이스		1	2	3	4	5
입력 항목						
입력 값	원인 1(교수)	F	T	F	T	F
	원인 2(학생)	T	F	F	F	T
	원인 3(#)	T	T	X	F	F
중간 노드	구성원	T	T	F	T	T
출력 값	출입 금지	F	F	T	F	F
	출입 가능	T	T	F	F	F
	비밀번호 오류	F	F	F	T	T

16. 구현 기반 테스트(화이트박스 테스트, 코드 기반 테스트)

- 구현 기반 테스트

 - = 화이트 박스 테스트 white box test

 - = 코드 기반 테스트 code based test

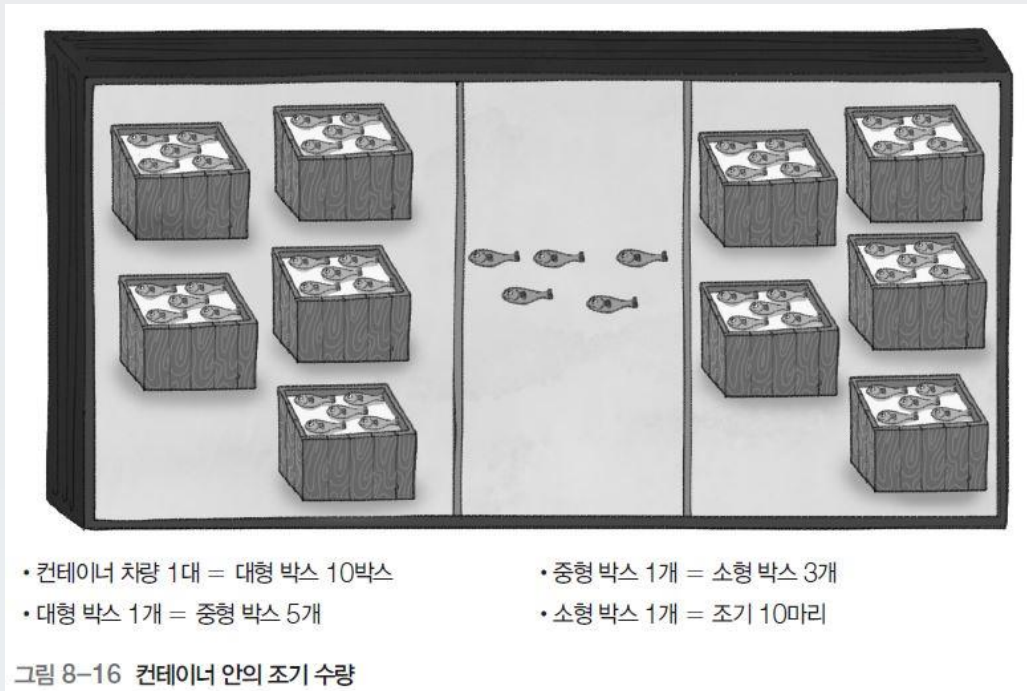
- 테스트 데이터 적합성 기준 test data adequacy criterion

 - = 테스트 데이터 생성 기준 test data generation criterion

 - 테스트에서 프로그램 코드의 가능한 경로를 모두 테스트할 수 없기 때문에 프로그램의 일부 경로만 정해 테스트해야 하는데 어떤 경로를 테스트 대상으로 선정할지 결정할 수 있는 기준

17. 화이트박스 테스트 절차

■ 테스트 데이터 적합성 기준 선정



18. 화이트박스 테스트 방법(1)

- 테스트 데이터 생성 test data generation
 - 명세나 원시 코드를 분석하여 선정된 기준을 만족하는 입력 데이터를 만든다.
- 테스트 실행
 - 프로그램 실행 → 실행 결과가 예상된 결과와 같은지 비교
- 화이트박스 테스트 방법

- 문장 검증 기준 statement coverage
- 분기 검증 기준 branch coverage
- 조건 검증 기준 condition coverage
- 분기/조건 검증 기준 branch/condition coverage
- 다중 조건 검증 기준 multiple condition coverage
- 기본 경로 테스트 basic path test

19. 화이트박스 테스트 방법(2)

- 방법 별로 복잡도와 소요 시간이 다르므로 테스트의 목적과 조건에 맞는 방법을 선택

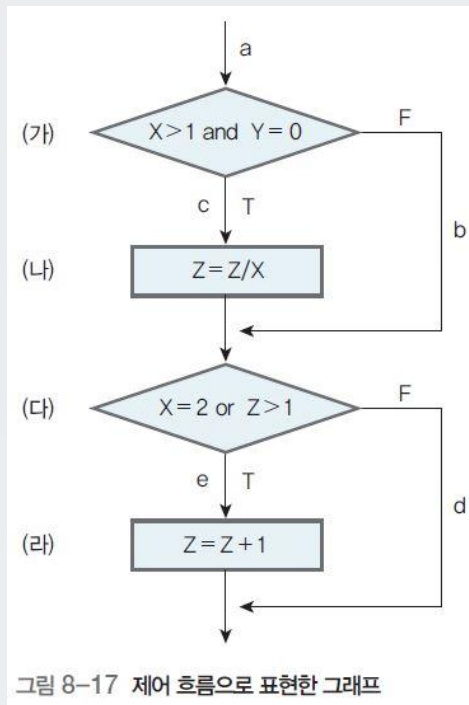
- 문장 검증 기준 statement coverage
- 분기 검증 기준 branch coverage
- 조건 검증 기준 condition coverage
- 분기/조건 검증 기준 branch/condition coverage
- 다중 조건 검증 기준 multiple condition coverage
- 기본 경로 테스트 basic path test

20. 문장 검증 기준(1)

■ 문장 검증 기준 statement coverage

- 프로그램 내의 모든 문장이 최소한 한 번은 실행될 수 있는 테스트 데이터를 갖는 테스트 케이스를 선정

원시 코드 → 제어 흐름 그래프



21. 문장 검증 기준(2)

■ 가능한 모든 경로 구함

- 프로그램 내의 모든 문장이 최소한 한 번은 실행될 수 있는 테스트 데이터를 갖는 테스트 케이스를 선정

표 8-12 가능한 모든 경로

번호	경로(가, 다)	가능 경로	만족 여부	이유
1	경로 1 (T, T)	a-c-e	만족	(가), (나), (다), (라) 문장을 모두 지나감.
2	경로 2 (T, F)	a-c-d	불만족	(라) 문장을 안 지나감.
3	경로 3 (F, T)	a-b-e	불만족	(나) 문장을 안 지나감.
4	경로 4 (F, F)	a-b-d	불만족	(나), (라) 문장을 안 지나감.

22. 문장 검증 기준(3)

- 모든 경로 중 문장 검증 기준을 만족하는 경로 선택

표 8-13 문장 검증 기준을 만족하는 경로

번호	경로(가, 다)	가능 경로	만족 여부	이유
1	경로 1 (T, T)	a-c-e	만족	(가), (나), (다), (라) 문장을 모두 지나감.

- 선택한 경로에 해당하는 테스트 데이터를 가지고 실행

표 8-14 테스트 데이터로 실행

번호	경로(가, 다)	테스트 데이터	가능 경로	출력 값	만족 여부
1	경로 1 (T, T)	X=2, Y=0, Z=3	a-c-e	2.5	만족

23. 문장 검증 기준 문제점(4)

■ 첫번째 조건문에서의 문제

- 원래는 or인데 실수로 and로 코딩한 경우 : 검증 기준 방법으로는 오류를 발견하지 못함
- → and인 경우, or인 경우 모두 (나) 문장을 지나가기 때문

and인 경우: 입력 값(X=2, Y=0, Z=3), 출력 값(Z=2.5)
or인 경우: 입력 값(X=2, Y=0, Z=3), 출력 값(Z=2.5)

■ 두 번째 조건문에서의 문제

- $Z > 1$ 식을 $Z > 0$ 을 잘못 코딩해도 오류를 발견하지 못함
- → or의 특성이 둘 중 하나만 만족하면 다른 조건식은 결과에 영향을 주지 않기 때문



해결 방법?

분기 검증 기준

조건문에 대해 T와 F가 적어도 한 번씩 수행할 수 있는 testcse 선정



24. 분기 검증 기준(1)

- 분기 검증 기준(branch coverage), 결정 검증 기준(decision coverage)
 - 조건문에 대해 T, F가 최소한 한 번은 실행되는 입력 데이터를 테스트 케이스로 사용
 - 분기 시점 또는 합류 위치에서 조건과 관련된 오류를 발견할 가능성이 높다.

① 원시 코드 → 제어 흐름 그래프

② 가능한 모든 경로 구함

표 8-15 가능한 모든 경로

번호	경로(가, 다)	가능 경로	만족 여부	이유
1	경로 1(T, T)	a-c-e	불만족	(F, F) 경로를 테스트 안 함.
2	경로 2(T, F)	a-c-d	불만족	(F, T) 경로를 테스트 안 함.
3	경로 3(F, T)	a-b-e	불만족	(T, F) 경로를 테스트 안 함.
4	경로 4(F, F)	a-b-d	불만족	(T, T) 경로를 테스트 안 함.

25. 분기 검증 기준(2)

- 모든 경로 중 분기 검증 기준을 만족하는 경로 선택
 - 네 개의 경로 중 하나만으로는 분기 검증 기준을 만족시키지 못함
 - 방법: 두 개의 경로를 묶어서 분기 검증 기준을 만족시킬 수 있는 경우를 찾는다.
 - $(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4) \rightarrow (1, 4)$ 또는 $(2, 3)$

표 8-16 1과 4의 테스트 케이스

번호	경로(가, 다)	가능 경로	이유
1	경로 1(T, T)	a-c-e	(T, T) 경로를 테스트함.
4	경로 4(F, F)	a-b-d	(F, F) 경로를 테스트함.

표 8-17 2와 3의 테스트 케이스

번호	경로(가, 다)	가능 경로	이유
2	경로 2(T, F)	a-c-d	(T, F) 경로를 테스트함.
3	경로 3(F, T)	a-b-e	(F, T) 경로를 테스트함.

26. 분기 검증 기준(3)

표 8-18 1, 4의 테스트 결과

번호	경로(가, 다)	테스트 데이터	가능 경로	출력 값	두 경로의 합
1	경로 1(T, T)	X=2, Y=0, Z=10	a-c-e	Z=6	만족
4	경로 4(F, F)	X=0, Y=0, Z=1	a-b-d	Z=1	

표 8-19 2와 3의 테스트 결과

번호	경로(가, 다)	테스트 데이터	가능 경로	출력 값	두 경로의 합
2	경로 2(T, F)	X=3, Y=0, Z=2	a-c-d	Z=1	만족
3	경로 3(F, T)	X=2, Y=1, Z=2	a-b-e	Z=2	

- 경로 1, 4 : $Z > 1$ 를 $Z < 1$ 로 코딩 실수 해도 그 오류를 발견 못함
- 이유 : 개별 조건식이 or로 연결되어 있어 둘 중 하나만 만족하면 두번째 식이 무엇이든 관계 없이 조건문의 결과 값에 영향을 주지 않기 때문



해결방법?

조건 검증 기준

조건문 내의 개별 조건식에 대하여 각각 T와 F인 경우를 최소한 한 번씩 수행



27. 조건 검증 기준(1)

```
IF(score>=90 and report>=90) THEN printf("A")
```

- 조건문만 고려한 ‘분기 검증 기준’의 문제(1)
 - 두 개별 조건식이 T일 때만 조건문이 T이고, 나머지는 두 조건식과 관계없이 모두 F
 - 이유: 두 개별 조건식이 and로 연결되어 있음
 - 개별 조건식이 (T, T)를 제외하고는 나머지 (T, F), (F, T), (F, F)는 모두 거짓이 되기 때문

표 8-20 and 연산자 사용 시 조건문의 결과

개별 조건식 A (score >= 90)	연산자	개별 조건식 B (report >= 90)	조건문 (전체 조건식)
T	and	T	T
T		F	F
F		T	F
F		F	F

28. 조건 검증 기준(2)

```
IF(score>=90 and report>=90) THEN printf("A")
```

- 조건문만 고려한 '분기 검증 기준'의 문제(2)

- 개별 조건식 A(score>=90)가 T이고, B(report >=90)가 F인 경우

개별 조건식 A: T에서 F로 바뀔

(F, T)에서 개별 조건식 B: T에서 F로 바뀔

(F, F)에서 개별 조건식 둘 중 하나가 T로 바뀔

모두 오류 발견 못함

- 이유: and의 특성이 하나라도 F이면 전체 조건식이 항상 F이기 때문



해결 방법?

조건 검증 기준

29. 조건 검증 기준(3)

■ 조건 검증 기준

- 두 개의 개별 조건식이 존재할 때 개별 조건식의 T와 F를 최소한 한 번은 테스트할 수 있도록 테스트 케이스 선정
- 분기 검증 기준에서 발견하지 못한 오류(개별 조건식에 존재하는 오류)를 발견할 수 있는 더 강력한 테스트

■ 조건 검증 기준의 테스트 케이스(4가지)

(가) : $(X > 1) = T$ and $(Y = 0) = T$

(가) : $(X > 1) = T$ and $(Y = 0) = F$

(가) : $(X > 1) = F$ and $(Y = 0) = T$

(가) : $(X > 1) = F$ and $(Y = 0) = F$

(가)의 개별 조건식을 만족하는 테스트 케이스

{ (T, T), (F, F)
(T, F), (F, T)

30. 조건 검증 기준(4)

조건 검증 기준의 테스트 케이스(4가지)

(다) : $(X=2) = T$ or $(Z>1) = T$

(다) : $(X=2) = T$ or $(Z>1) = F$

(다) : $(X=2) = F$ or $(Z>1) = T$

(다) : $(X=2) = F$ or $(Z>1) = F$

(다)의 개별 조건식을 만족하는 테스트 케이스

(T, T), (F, F)

(T, F), (F, T)

표 8-21 [그림 8-17]에 대한 테스트 케이스

1	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	T	T	T
	F	F	F	F

2	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	T	T	F
	F	F	F	T

3	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	T	F	T
	F	F	T	F

4	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	T	F	F
	F	F	T	T

5	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	F	T	T
	F	T	F	F

6	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	F	T	F
	F	T	F	T

7	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	F	F	T
	F	T	T	F

8	(가)		(다)	
	개별 조건식		개별 조건식	
	A	B	A	B
	T	F	F	F
	F	T	T	T

31. 조건 검증 기준(5)

표 8-22 [그림 8-17]에 대한 테스트 케이스

번호	경로			개별 조건식	두 경로의 합	테스트 케이스	전체 조건식	
	(가)	(다)						
1	경로 1	T, T	T, T	불만족	만족	적합	(가)T, (다)T	만족
	경로 2	F, F	F, F	불만족			(가)F, (다)F	
2	경로 3	T, T	T, F	불만족	만족	적합	(가)T, (다)T	불만족/ (다)F가 없음
	경로 4	F, F	F, T	불만족			(가)F, (다)T	
3	경로 5	T, T	F, T	불만족	만족	제외(경로 3, 4 와 중복)	(가)T, (다)T	X
	경로 6	F, F	T, F	불만족			(가)F, (다)T	
4	경로 7	T, T	F, F	불만족	만족	적합	(가)T, (다)F	만족
	경로 8	F, F	T, T	불만족			(가)F, (다)T	
5	경로 9	T, F	T, T	불만족	만족	적합	(가)F, (다)T	불만족/ (가)T가 없음
	경로 10	F, T	F, F	불만족			(가)F, (다)F	
6	경로 11	T, F	T, F	불만족	만족	적합	(가)F, (다)T	불만족/(가)T, (다)F가 없음
	경로 12	F, T	F, T	불만족			(가)F, (다)T	
7	경로 13	T, F	F, T	불만족	만족	제외(경로 11, 12와 중복)	(가)T, (다)T	X
	경로 14	F, T	F, F	불만족			(가)T, (다)T	
8	경로 15	T, F	F, F	불만족	만족	적합	(가)T, (다)T	불만족/(가), (다)F가 없음
	경로 16	F, T	T, T	불만족			(가)T, (다)T	

32. 분기/조건 검증 기준(1)

- 분기/조건 검증 기준branch/condition coverage
 - 개별 조건식을 모두 만족하면서 전체 조건식도 만족하는 테스트 케이스

표 8-23 분기/조건 검증 기준의 테스트 케이스

번호	경로			개별 조건식	두 경로의 합	전체 조건식	
		(가)	(다)				
1	경로 1	T, T	T, T	불만족	만족	(가)T, (다)T	만족
	경로 2	F, F	F, F	불만족		(가)F, (다)F	
4	경로 7	T, T	T, F	불만족	만족	(가)T, (다)F	만족
	경로 8	F, F	F, T	불만족		(가)F, (다)T	

33. 분기/조건 검증 기준(2)

- 첫 번째 조건문에서 문제점
 - $(X > 1)$ 가 F이면 $(Y = 0)$ 의 결과와 관계없이 조건문이 $F \rightarrow (Z = Z/X)$ 문장을 수행 안 함
∴ (가)의 개별 조건식 $(Y = 0)$ 에서 오류 발생 시 발견 못함
∴ 두 개의 개별식이 and로 연결 \rightarrow ∴ 개별 조건식 하나만 F이면 조건문은 무조건 F
- 두 번째 조건문에서 문제점
 - $(X = 2)$ 가 T이면 $(Z > 1)$ 와 관계없이 문장 $(Z = Z + 1)$ 을 반드시 수행
∴ 두 개의 개별식이 or로 연결되었기 때문
∴ (다)의 개별 조건식 $(Z > 1)$ 에서 오류가 발생해도 이를 발견 못함
- 마스크(mask)
 - 어떤 개별 조건식이 다른 개별 조건식의 결과와 상관없이 이미 결정되어지는 것

34. 다중 조건 검증 기준(1)

- and로 연결된 개별 조건식에서의 마스크 문제 해결 방법
[문제] 두 식 중 하나가 F인 경우 나머지 식이 F이든 T이든 상관없이 결과가 F라는 것
[해결 방법] 나머지 식에서 T와 F인 경우를 각각 하나씩 추가하여 테스트 케이스 선정
- or로 연결된 개별 조건식에서의 마스크 문제 해결 방법
[문제] 두 식 중 하나가 T인 경우 나머지 식은 F이든 T이든 상관없이 결과가 T라는 것
[해결 방법] 나머지 식에서 T인 경우와 F인 경우를 하나씩 추가하여 테스트 케이스 선정
- 다중 조건 검증 기준^{multiple condition coverage}
 - 마스크 문제까지 해결한 테스트 케이스에 해당하는 테스트 데이터를 생성하는 기준

35. 다중 조건 검증 기준(2)

다중 조건 검증 기준의 테스트 케이스

표 8-24 다중 조건 검증 기준의 테스트 케이스

번호	경로		분기/조건 검증 기준		다중 조건 검증 기준
	(가)	(다)	두 경로의 합	전체 조건식	
1	경로 1	T, T	T, T	만족	만족
	경로 1	T, T	T, F		
	경로 2	F, F	F, F		
	경로 2	F, T	F, F		
4	경로 7	T, T	F, F	만족	만족
	경로 8	F, F	T, T		
	경로 8	F, F	T, F		
	경로 8	F, T	T, T		
	경로 8	F, T	T, F		

경로 1 : or가 T인 경우에 해당

경로 2 : and가 F인 경우에 해당

경로 7 : 해당 없음

경로 8 : and가 F인 경우, or가 T인 경우 모두 해당

36. 기본 경로 테스트(1)

- 기본 경로 테스트 basic path test
 - 원시 코드의 독립적인 경로가 최소한 한 번은 실행되는 테스트 케이스를 찾아 테스트
 - 목적: 원시코드의 독립적인 경로를 모두 수행하는 것
- 기본 경로 테스트의 수행 절차

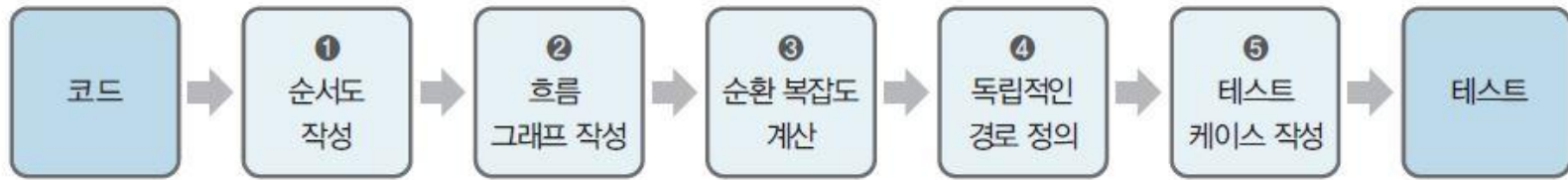


그림 8-18 기본 경로 테스트 수행 절차

37. 기본 경로 테스트(2)

- 순서도 작성
 - 문장statement, 조건문if then else, 반복문for과 같은 3가지의 기본 구조로 표현

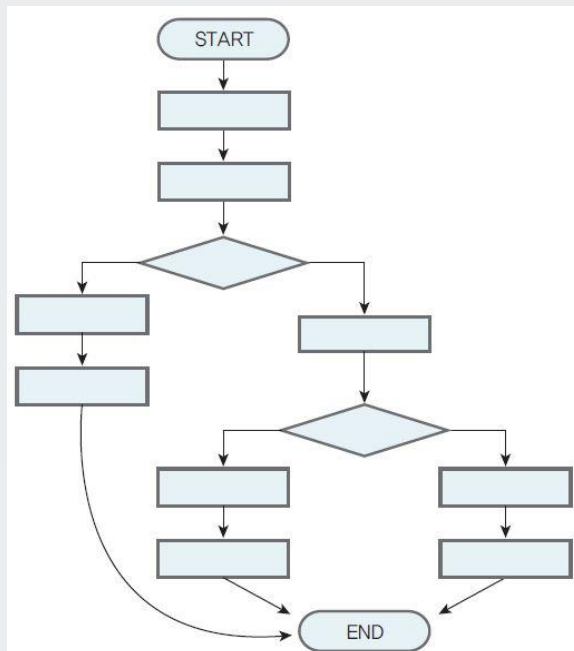


그림 8-19 순서도의 표현

38. 기본 경로 테스트(3)

■ 흐름 그래프 작성

- 순서 구조, 선택 구조와 같은 표기법을 사용하여 원시 코드 흐름 그래프를 표현

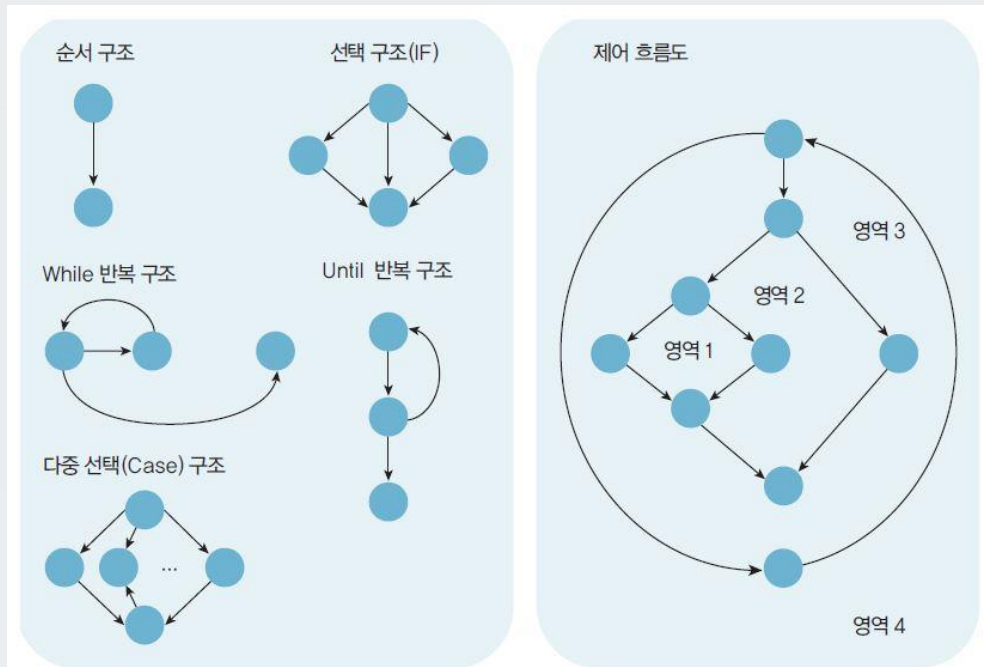
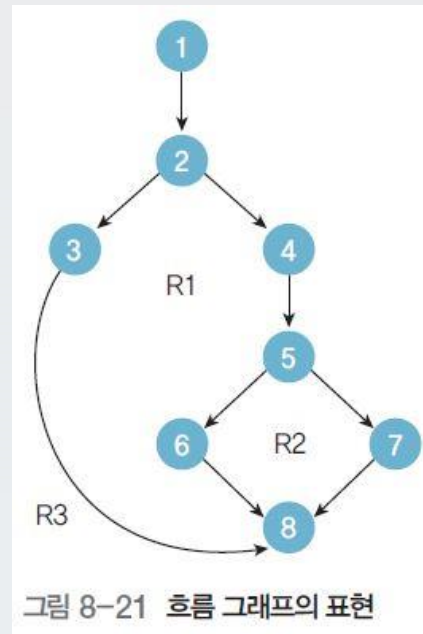


그림 8-20 흐름 그래프에서 사용되는 표기



39. 기본 경로 테스트(4)

■ 순환 복잡도 계산

- 순환 복잡도 CC: Cyclomatic Complexity

- 매케이브가 정의한 메트릭으로 원시 코드의 복잡도를 정량적으로 평가하는 방법
- 원시 코드가 얼마나 복잡한지를 알아보기 위한 것
- 얼마나 많은 논리적인 경로를 가지고 있는지 계산한 값

■ 순환 복잡도 계산 공식

- CC5R의 수53
- CC5E2N1259281253
- CC5P11521153

[순환 복잡도 공식]

- CC5R의 수
- CC5E2N12
- CC5P11
- ✓ R(Region) : 화살표와 노드로 둘러싸인 구역 외부 구역도 하나의 영역으로 간주함.
- ✓ E(Edge) : 화살표 수
- ✓ N(Node) : 노드 수
- ✓ P(Predicate) : 분기 노드 수

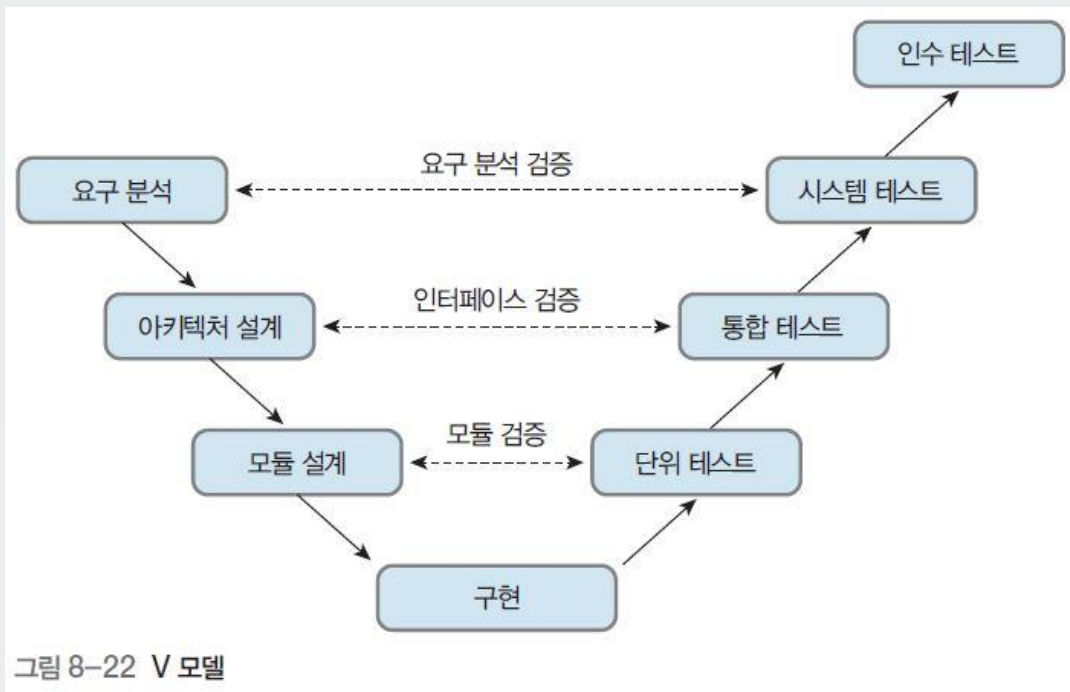
40. 기본 경로 테스트(5)

- 독립적 경로 정의
 - 순환 복잡도가 3이므로 독립적인 경로는 3개
 - 경로 1: 1-2-3-8
 - 경로 2: 1-2-4-5-6-8
 - 경로 3: 1-2-4-5-7-8
- 테스트 케이스 작성
 - 테스트 케이스는 3개의 경로에 해당하는 테스트 데이터를 생성하면 된다.

41. 소프트웨어 개발 단계에 따른 테스트

■ V 모델

- 소프트웨어 개발 단계의 순서와 짝을 이루어 테스트를 진행해나가는 방법
- 프로젝트 초기 단계부터 테스트 계획을 세우고, 테스트 설계 과정이 함께 진행



42. 단위 테스트(1)

- 단위 테스트, 모듈 테스트(module test)
 - 프로그램의 기본 단위인 모듈을 테스트
 - 모듈 개발 완료한 후 명세서의 내용대로 정확히 구현되었는지를 테스트
(즉) 개별 모듈이 제대로 구현되어 정해진 기능을 정확히 수행하는지를 테스트
 - 프로그램의 기본 단위인 모듈을 테스트
- 단위 테스트 수행 후 발견되는 오류
 - 잘못 사용한 자료형
 - 잘못된 논리 연산자
 - 알고리즘 오류에 따른 원치 않는 결과
 - 틀린 계산 수식에 의한 잘못된 결과
 - 탈출구가 없는 반복문의 사용

43. 단위 테스트(2)

- 모듈 테스트 시 상위/하위 모듈이 개발 안된 경우
 - 가상의 상위나 하위 모듈을 만들어 사용
 - 테스트 드라이버^{test driver}
 - 상위 모듈의 역할을 하는 가상의 모듈, 테스트할 모듈 호출
 - 필요한 데이터를 인자를 통하여 넘겨주고, 테스트가 완료된 후 그 결과 값을 받는 역할
 - 테스트 스텝^{stub}
 - 하위 모듈의 역할
 - 테스트할 모듈이 호출할 때 인자를 통해 받은 값을 가지고 수행 후 결과를 테스트할 모듈에 넘겨주는 역할



그림 8-23 드라이버/스텝 모듈과 테스트 대상 모듈의 관계

44. 통합 테스트(1)

- 통합 테스트 integration test
 - 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생할 수 있는 오류를 찾는 테스트
 - '모듈 간의 상호작용이 정상적으로 수행되는가' 테스트
 - 모듈 사이의 인터페이스 오류는 없는지, 모듈이 올바르게 연계되어 동작하고 있는지 체크
- 모듈 통합 방법에 따른 분류
 - 한꺼번에 하는 방법: big-bang 테스트
 - 점진적으로 하는 방법: 하향식 기법, 상향식 기법
- Big-bang 테스트
 - 단위테스트가 끝난 모듈을 한꺼번에 결합하여 수행하는 방식
 - 소규모 프로그램이나 프로그램의 일부를 대상으로 하는 경우에 적합
 - 오류 발생 시 어떤 모듈에서 오류가 존재하는지, 그 원인이 무엇인지 찾기가 어려움

45. 통합 테스트(2)

■ 점진적 모듈 통합 방법 : 하향식^{top-down} 기법

- 모듈의 계층 구조에서 맨 상위의 모듈부터 시작하여 점차 하위 모듈 방향으로 통합
- 모듈의 구성
 - 상위 모듈: 시스템 전체의 흐름 관장
 - 하위 모듈: 각 기능을 구현
- 장점
 - 프로그램 전체에 영향을 줄 수 있는 오류를 일찍 발견하기가 쉽다.
- 단점
 - 하위 모듈이 임시로 만든 스텝들로 대체되어 결과가 완전하지 않을 수도 있다.
 - 스텝 수가 많으면 스텝을 만드는 데 시간과 노력이 많이 들 수 있다.

∴ 모듈 간의 인터페이스와 시스템의 동작이 정상적으로 잘되고 있는지를 빨리 파악하고자 할 때 유용

46. 통합 테스트(3)

① 넓이 우선 breadth first 방식

- 같은 행에서는 옆으로 가며 통합
- $(A, B) \rightarrow (A, C) \rightarrow (A, D) \rightarrow (A, B, E) \rightarrow (A, B, F) \rightarrow (A, C, G)$

② 깊이 우선 방식 depth first 방식

- 같은 행에서는 아래로 가며 통합
- $(A, B) \rightarrow (A, B, E) \rightarrow (A, B, F) \rightarrow (A, C) \rightarrow (A, C, G) \rightarrow (A, D)$

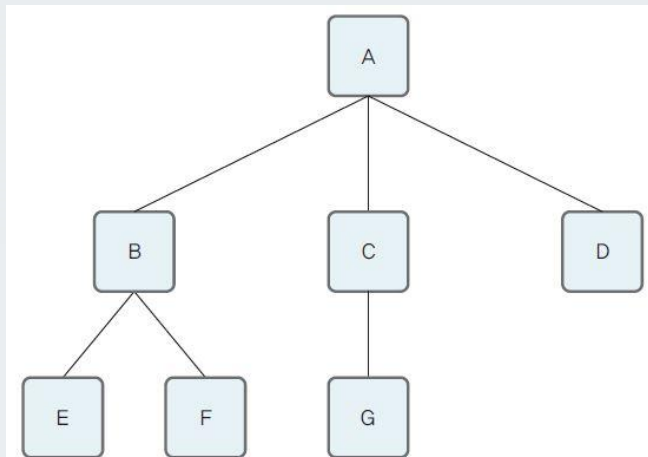


그림 8-24 점진적 모듈 통합 방법을 설명하기 위한 모듈 구성도

47. 통합 테스트(4)

■ 점진적 모듈 통합 방법 : 상향식^{bottom-down} 기법

- 가장 말단에 있는 최하위 모듈부터 테스트
- 상위 모듈의 역할을 하는 테스트 드라이버가 필요
- 드라이버: 하위 모듈을 순서에 맞게 호출하고, 호출할 때 필요한 매개 변수를 제공하며, 반환 값을 전달하는 역할
- 순서
 - 우선 가장 말단(레벨 3)에 있는 모듈 E와 F를 모듈 B에 통합하여 테스트
 - 그 다음 모듈 G를 모듈 C에 통합하여 테스트
 - 마지막으로 모듈 B, C, D를 모듈 A에 통합하여 테스트
- 장점
 - 최하위 모듈들을 개별적으로 병행하여 테스트 → 하위에 있는 모듈들을 충분히 테스트 가능
 - 정밀한 계산이나 데이터 처리가 요구되는 시스템 같은 경우에 유용
- 단점
 - 상위 모듈에 오류가 발견되면 그 모듈과 관련된 하위의 모듈을 다시 테스트해야 함

49. 시스템 테스트

■ 시스템^{system} 테스트

- 시스템 전체가 정상적으로 작동하는지를 체크
- 모듈이 모두 통합된 후 사용자의 요구 사항들을 만족하는지 테스트
- 사용자에게 개발된 시스템을 전달하기 전에 개발자가 진행하는 마지막 테스트
- V & V에서 확인^{verification}에 해당
- 실제 사용 환경과 유사하게 테스트 환경을 만들어놓고 요구 분석 명세서에 명시한 기능적 요구 사항과 비기능적 요구 사항을 충족하는지 테스트
- 주로 부하를 주는 상황에서 수행하고, 비기능적 테스트를 중심으로 수행

50. 인수 테스트(1)

■ 인수 테스트 acceptance test

- 시스템이 예상대로 동작하는지 확인하고, 요구 사항에 맞는지 확신하기 위해 하는 테스트
- 시스템을 인수하기 전 요구 분석 명세서에 명시된 대로 모두 충족시키는지를 사용자가 테스트
- 목적: 사용자 주도로 이루어지며, 오류 발견보다는 제품의 출시 여부를 판단하는 것
- 인수 테스트 결과: 시스템을 출시할지, 출시 시기를 늦추더라도 보완할지 결정
- V & V의 검증(validation)에 해당

51. 인수 테스트(2)

① 알파 테스트alpha test

- 내부 필드 테스트
- 베타 테스트 개발자 환경에서 사용 → 오류와 사용상의 문제점 파악

② 베타 테스트beta test

- 알파 테스트 후 시장 출시 전 시장의 피드백을 얻기 위한 목적으로 테스트
- 특정 사용자가 미리 사용 → 문제점, 오류 발견 → 개발자에게 알려줌
- 개발자: 베타 테스트를 통해 보고된 문제점 수정 → 제품 출시

52. 회귀 테스트

- **확정 테스트**confirmation test

- 원시 코드의 결함을 수정한 후 제대로 수정되었는지 확인하는 테스트

- **회귀 테스트**regression test

- 한 모듈의 수정이 다른 부분에 영향을 끼칠 수도 있다고 생각하여 수정된 모듈뿐 아니라 관련된 모듈까지 문제가 없는지 테스트

- ① **수정을 위한 회귀 테스트**corrective regression test

- 모든 테스트를 완료하여 사용자에게 전달하기 전에 테스트 과정에서 미처 발견하지 못한 오류를 찾아 수정한 후 다시 테스트

- ② **점진적 회귀 테스트**progressive regression test

- 사용 중에 일부 기능을 추가하여 새로운 버전을 만들고, 이 새 버전을 다시 테스트

다음 시간

기본 자바 소개 2



송실사이버대학교

송실사이버대학교의 강의콘텐츠는
저작권법에 의하여 보호를 받는바, 무단
전재, 배포, 전송, 대여 등을 금합니다.

* 사용서체 : 나눔글꼴