Computational Thinking

2017-2학기

9장-정확하게 만들기

박종혁 교수 (컴퓨터공학과)

jhpark1@seoultech.ac.kr

http://www.parkjonghyuk.net

서울과학기술대학교 SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

학습개요

- 9장 정확하게 만들기
  1. "컴퓨터 오류"란 보통은.
  2. 소프트웨어 정확성
  3. 검증
  4. 소프트웨어 테스팅
  5. 화이트 박스 테스트
  6. 균등 분할을 통한 블랙박스 테스트
  7. 경계값 분석
- 조별과제
  - 다음 주 강의 시간 발표

학습목표

- 많은, 대부분은 아니지만, 컴퓨터 오류라고 불리는 것들은 사실은 사람에 의한 오류이며 보통은 데이터를 잘못 입력해서 생기는 오류라는 것을 이해한다.
- 명세서에서 정확한 것이 무엇인가가 정의되지 않으면 정확성은 불가능하다는 것을 이해한다.
- 검증과 확인 사이의 차이점과 그들이 중요한 이유를 설명할 수 있다.
- 소프트웨어의 정확성을 증명하는 것은 가능하지만 이러한 증명은 비용과 복잡성 때문에 극히 드물게 사용되고 있다는 점을 깨닫는다.
- 정확성과 관련해서 소프트웨어 테스트의 한계를 안다.

9.1 "컴퓨터 오류"란 보통은...

- 컴퓨터가 정확하게 동작하지 않는 것을 통틀어 말함
- 대부분의 사람의 실수에 의한 것들
- 따라서 사람이 실수를 덜 한다면 오류도 덜 나타나게 됨. 하지만 사람은 항상 실수를 하며 더 많은 자료의 값을 입력하면 할수록 더 많은 실수를 하게 됨.
- 실수를 덜 하도록 만들기 보다는 실수를 했을 경우에 검증 시스템을 통해 실제 시스템에 반영이 되지 않도록 만드는 것이 더 중요함

9.1 "컴퓨터 오류"란 보통은... (컴퓨터 오류의 원인)

- **부정확한 자료 입력**
  - 은행 계좌 이체 금액 잘 못 입력, 주민번호 실수 입력
- **시스템 이해 부족**
  - 이메일 답장 할 때 전체 답장, 브라우저 자동 완성기능
- **컴퓨터 출력 결과의 부정확한 해석**
  - 구매가격 잘 못 읽어 구매, 네비게이터 정보 잘 못 이해
- **물리적 피해**
  - 물리적 파손, 침수, 화재
- **하드웨어 고장**
  - 마우스 고장, DVD 플레이어 고장
- **소프트웨어 결함**
  - 프로그램 버그
- **보안 결함**
  - 외부 해킹

1. **소프트웨어가 고객의 요망에 맞게 동작한다.**
   - 확인(Validation) 프로세스를 통해 검사

2. **소프트웨어가 작성된 요구사항 명세서에 맞게 동작한다.**
   - 검증(Verification) 프로세스를 통해 검사

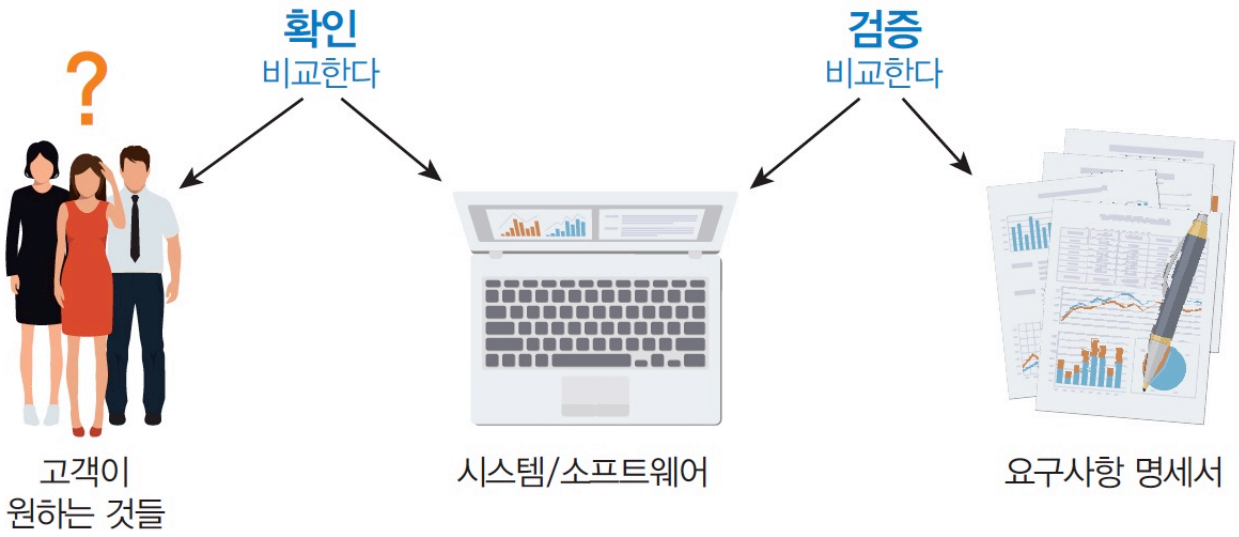

그림 9.2 두 종류의 정확성과 그들이 보장하기 위한 방법

❖ 확인 프로세스

- 고객이 원하는 대로 소프트웨어가 동작하는 지를 점검하는 프로세스

❖ 확인 프로세스 기법들

- 베타 테스트
  - 선택된 사용자들이 시스템을 사용해 보고 의견을 말하게 함
- 사용성 테스트
  - 선택된 개개인들이 시스템의 외관과 느낌을 탐색하게 함
- 인수 테스트
  - 고객이 최종 구매에 앞서 시스템을 확인함

❖ **소프트웨어 요구사항이 명시된 명세서에 규정된 대로 소프트웨어가 동작하는 지를 점검하는 프로세스**

❖ **요구사항이 명확하고 일관성 있고 완전해야 검증 프로세스가 제대로 진행됨**

- 요구사항 S1) 프로그램이 안전해야 한다.
  - 검증 가능하지 않음
- 요구사항 S2) 모든 사용자는 사용자 이름과 패스워드를 가지고 로그인을 해야 한다.
  - 검증 가능함
- 요구사항 E1) 프로그램은 빨라야 한다.
  - 검증 가능하지 않음
- 요구사항 E2) 0.5초 내로 시스템은 반응해야 한다.
  - 검증 가능함

❖ 요구사항의 분류

- 기능적 요구사항
  - 소프트웨어가 '무엇을 하는 가'를 정의
- 비기능적 요구사항
  - 소프트웨어가 어떻게 잘 동작하는 가를 정의
  - 성능, 보안, 이동성, 신뢰성, 안전, 문서화, 물품 인도, 사용자 친화력 같은 부분을 기술
  - 검증이 가능하지 않은 S1과 E1의 모두 비기능적 요구사항

❖ 요구사항 작성시 모든 요구사항이 검증이 가능하도록 주의를 기울여야 함.

❖ 소프트웨어 테스팅

- 프로그램을 실행하고 명세서에 기록된 내용과 일치하는 지를 비교함
- 소프트웨어 테스팅은 정확성을 보증하지는 않음
  - F22 랩터 통신시스템의 고장 사례
  - 다양한 사용 환경에 대한 모든 경우의 수를 테스트 할 수 없기 때문
  - 따라서 모든 경우에 대한 테스팅 샘플을 만들어 테스트를 함
  - 결국 어떤 테스팅 샘플을 만드느냐가 소프트웨어 테스팅 성능에 중요한 영향을 미침
- 화이트박스 테스트와 블랙박스 테스트로 나뉨

❖ 테스트를 위해서는 테스트 케이스를 작성해야 한다

- 입력조건과 예상 결과를 작성
- 입력조건은 변수이름, 예상결과는 입력에 따른 프로그램 행동을 의미

정확한 User name 과 비밀번호
User Name :
Password :

테스트 케이스	예상 결과
User Name : Armstrong Password : 2Dogs@Home	로그인 성공
User Name : Armstrong Password : 2Dogs1Home	로그인 실패
User Name : Armstrang Password : 2Dogs@Home	로그인 실패

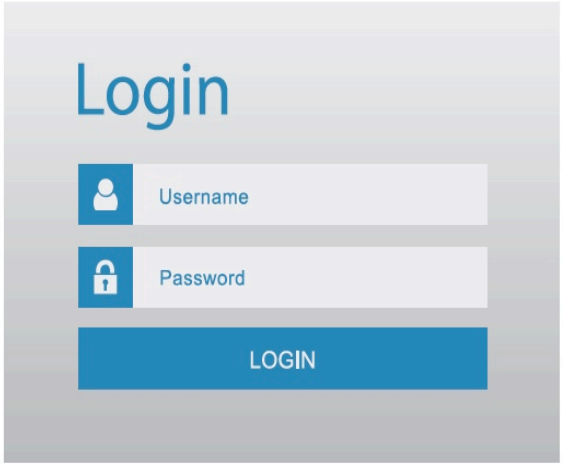

그림 9.3 로그인 패널

❖ **테스트 결과가 안전하게 나온다고 해서 시스템이 안전하다고 보장할 수 없음**

- 입력된 값에만 안전한 결과가 나오는 것이며 다른 입력 값에는 잘못된 행동을 할 가능성이 있음.

❖ **모든 입력 값의 조합으로 테스트 할 수 없음**

- 너무 많은 경우의 수가 존재

❖ **테스트 샘플이 중요함**

- 모든 입력 경우를 대표할 수 있는 샘플을 선택하여 테스트를 진행해야 함
- 여론조사와 비슷: 여론 조사의 경우 조사 샘플이 모집단을 정확히 대표할 수록 정확한 값이 나옴

❖ 여러 개의 테스트 케이스를 사용해야 함

❖ 소프트웨어의 정확성을 확신할 수 있도록 조심스럽게 선택되어야 함

❖ 테스트 스위트
- 테스트 케이스의 집합

❖ 효과적인 테스트 스위트는 정확성을 보장하지는 못하지만 많은 소프트웨어 오류를 잡아낼 수 있음

❖ **테스트 케이스는 소프트웨어 개발과정의 단계마다 구성되어 실행되어야 함**
- 모듈 개발 단계, 서브 시스템 개발 단계, 전체 시스템 개발 단계

❖ **테스트 케이스는 소프트웨어 개발 중에 반복적으로 실행됨**
- 새로운 기능을 추가하거나 프로그램을 수정할 때마다 회귀 테스트를 실행해야 함.

❖ **소프트웨어 품질관리팀에서 전문적으로 수행**

❖ **테스트의 종류**
- 화이트박스 테스트
- 블랙박스 테스트

❖ 프로그램의 내부 구조를 아는 경우에 시행하는 테스트

- 구조 테스트
- 프로그래머가 직접 테스트 수행
- 경로 테스트가 대표적인 화이트박스 테스팅

❖ 화이트 박스 테스트 스위트

- 명령문 적용범위라고 불림
- 프로그램의 모든 명령어가 테스트 스위트의 적어도 하나의 테스트 케이스에 의해 최소 한번은 실행되어야 함

❖ 경로 테스트

- 활동 다이어그램 또는 프로그램 코드에서 프로그램 수행 경로 도출
- 각 경로마다 테스트 케이스 작성 후 테스트
- 테스트 결과와 예상 결과를 비교, 소프트웨어의 정확성을 판단

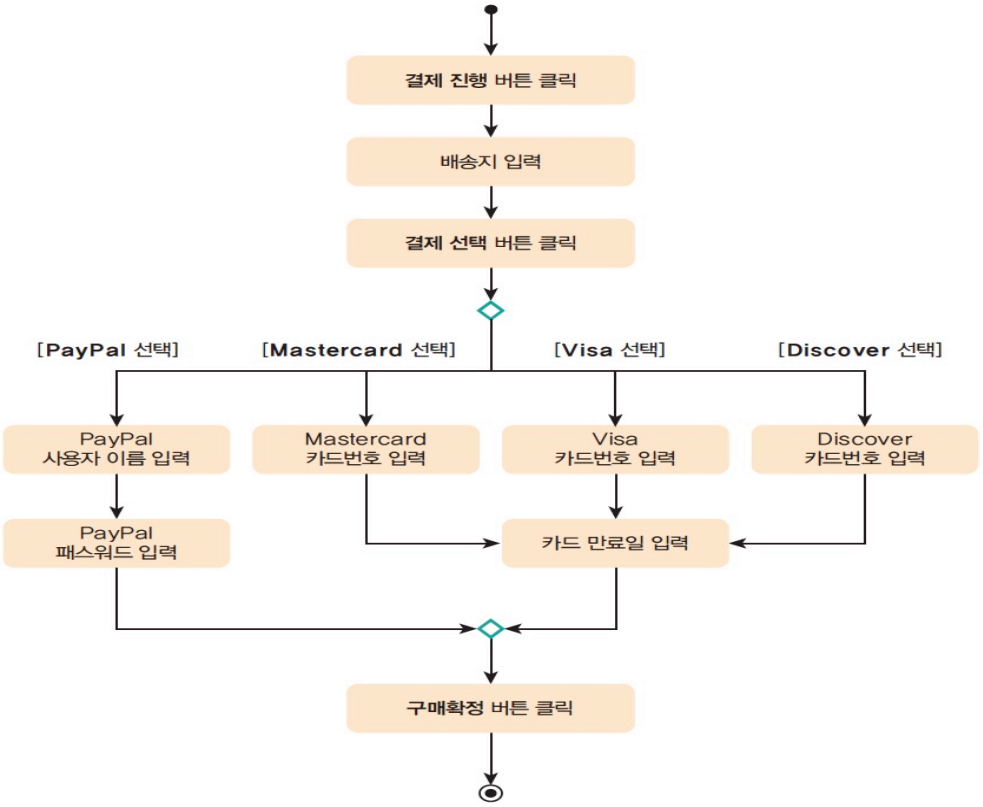

그림 9.5 온라인 구매 결제에 대한 활동 다이어그램

- 테스트 케이스는 모든 가능한 경로를 테스트 할 수 있도록 구성되어야 함
- Paypal 과 3개의 서로 다른 신용카드 결제에 대해 테스트 케이스가 작성되어 테스트가 실행되어야 함

9.5 화이트박스 테스팅 (경로테스트)

❖ 테스트 케이스는 19개의 모든 활동들이 실행 될 수 있도록 구성되어야 함.

❖ 적절한 입력 조건을 통해 19개의 모든 활동이 실행 될 수 있도록 테스트 케이스를 구성 할 수 있음.

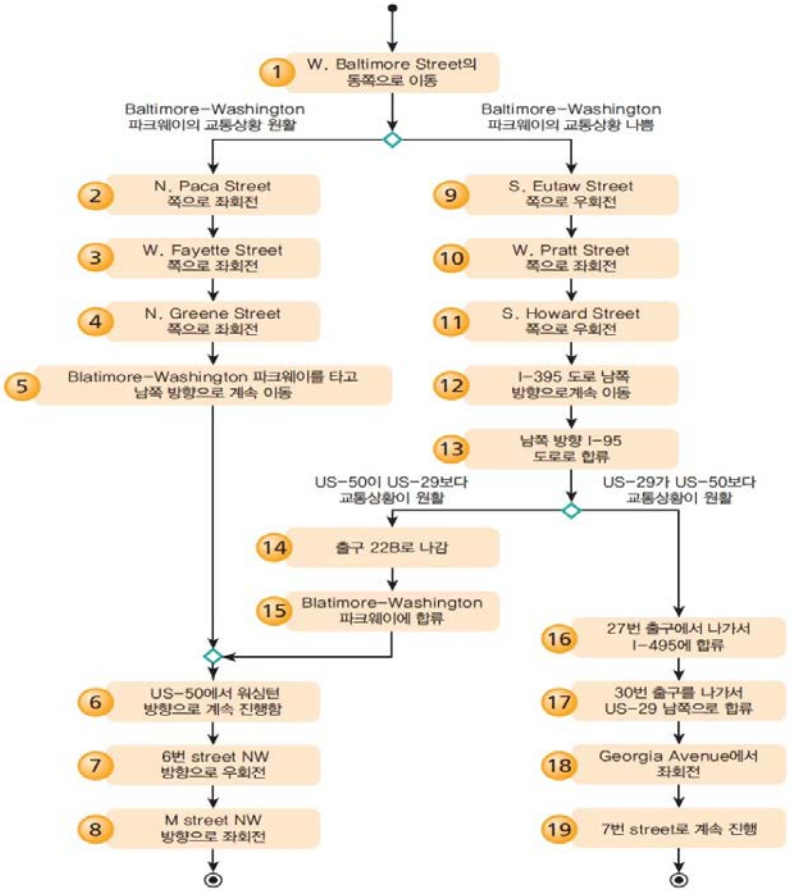

그림 9.6 매릴랜드의 볼티모어에서 워싱턴 시로 여행을 하는 활동 다이어그램

테스트 케이스	입력 조건	기대 행위
볼티모어 파크웨이	볼티모어 파크웨이 교통상황 좋음	워싱턴 시 도착
US-50	볼티모어 파크웨이 교통체증 중이며 US-50이 US-29보다 교통상황이 좋음	워싱턴 시 도착
US-29	볼티모어 파크웨이 교통체증 중이며 US-50이 US-29보다 교통상황이 안 좋음	워싱턴 시 도착

그림 9.7 그림 9.6에 대한 완전한 경로 적용범위 테스트 스위트

Test Case	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Paypal	✓	✓	✓	✓	✓	✓	✓	✓											
US-50	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
US-29	✓								✓	✓	✓	✓	✓			✓	✓	✓	✓

그림 9.8 그림 9.6에 대한 명령문 적용범위를 보장하기

가능한 경로

테스트 케이스 1) 1-2-3-5
테스트 케이스 2) 1-2-3-4-5
테스트 케이스 3) 1-2-3-4-4-5
테스트 케이스 4) 1-2-3-4-4-4-5

각 테스트 케이스 별로 해당하는 입력 값들을 정하고 테스트 수행

테스트 케이스 1) 전원버튼-앱버튼-5-녹색버튼
테스트 케이스 2) 전원버튼-앱버튼-5-3-녹색버튼
…

테스트 수행 후 나오는 결과가 예상하는 내용과 같은지 비교

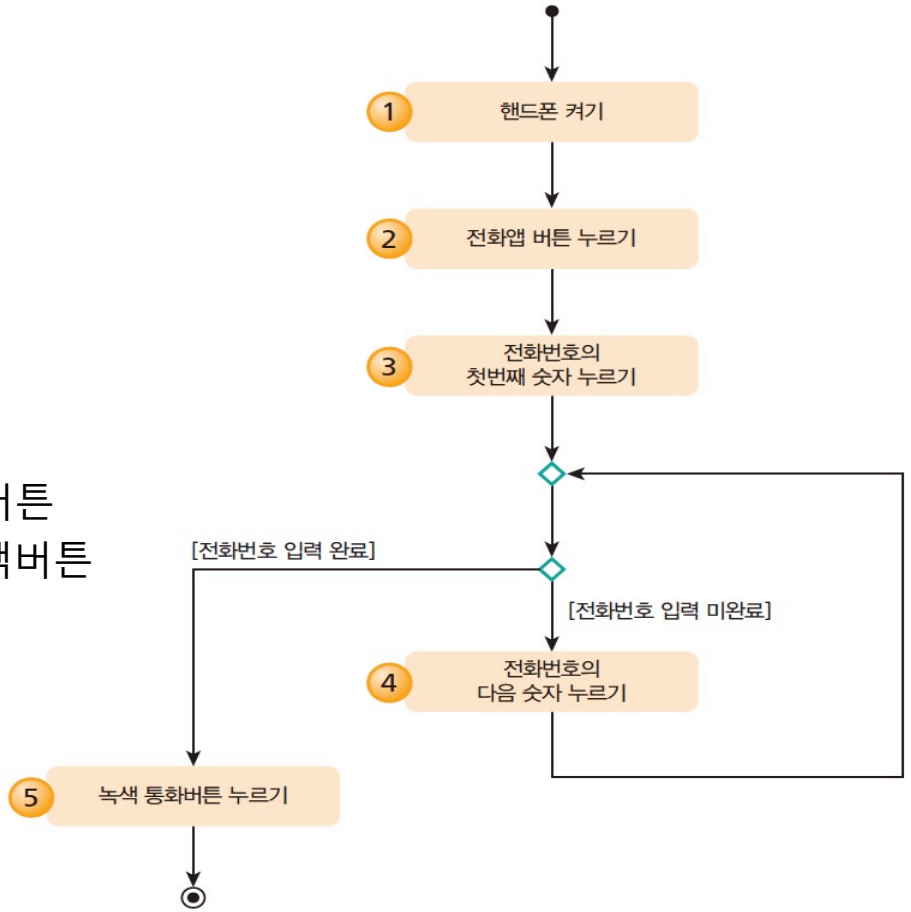

그림 9.9 전화를 거는 활동 다이어그램

❖ 소프트웨어에 입력되는 경우들을 균등하게 여러 그룹으로 나누어 각 그룹의 대표 값을 입력하여 테스팅 함

그림 9.11 피자 자르기는 균등 분할과 같다.

❖ 블랙박스 테스트

- 장점
  - 프로그래밍 경험이 거의 또는 전혀 필요 없음
  - EX) 주사위 게임 '크랩' (그림 9.10 참조)
  - 모든 요구사항은 명세서로 주어짐

일련번호: C1
이름: 컴 아웃(Come Out) 주사위 던지기
동작: 크랩게임은 두 개의 주사위를 한번 이상 던지는 것으로 구성되는 게임이다. 게임을 시작할 때 주사위를 던지는 것을 컴 아웃(Come Out) 주사위던지기라고 한다. 만약 주사위 결과가 2, 3, 12 인 경우 크랩 아웃에 의해 플레이어는 지게 된다. 만약 주사위 결과가 7 또는 11 인 경우 사용자는 내츄럴 위너(natural winner)가 된다. 만약 사용자가 다른 결과를 내게 된다면(예를 들면 4, 5, 6, 8, 9, 10), 이 주사위는 포인트가 되며 다음 주사위 던지기는 포인트 롤(point roll)이 된다.

일련번호: C2 :
이름: 포인트 롤
동작: 플레이어는 주사위를 다시 던진다. 만약 두 주사위의 합이 7이라면 플레이어는 세븐 아웃에 의해 게임을 지게 된다. 만약 주사위의 합이 포인트(컴 아웃 주사위 던지기에서 던졌던 주사위의 합)와 같다면, 플레이어는 포인트를 맞추었으므로 게임에서 이기게 된다. 만약 두 주사위의 합이 7이나 포인트와 다른 값이라면 포인트 롤은 계속 반복된다.

일련번호: C3
이름: 플레이 계속
동작: 크랩 아웃, 당연 승리, 세븐 아웃, 내츄럴 위너, 포인트 맞추기가 나오면, 플레이어는 새로운 게임을 시작할 수 있다(새로운 게임을 시작하는 내용은 요구사항 C1을 참조한다).

그림 9.10 크랩 게임의 요구사항들

- 차이점
  - 테스트 선정 방식에 따라 달라짐
  - 모든 테스트는 소프트웨어 요구사항에 따라서 작성됨
  - 명령어 / 경로 적용범위와 같은 기법들을 사용할 수 없음

❖ 자동차 안정성 테스트

- 경우의 수
  - 차가 움직이지 않음
  - 차가 움직임
  - 차가 최대 속력으로 움직임

테스트 케이스	입력 조건	예상 결과
차가 움직이지 않음	차량 속도 = 0	차가 움직이지 않음
차가 움직임	차량 속도 = 60 km/h	차가 뒤집히지 않음
차가 최대 속력으로 움직임	차량 속도 = 180 km/h	차가 뒤집히지 않음

❖ **분할된 입력 그룹에 따라 각 그룹의 다음 값들을 테스트**

- 경계선에 있는 값을 입력으로 하는 테스트
- 경계선 안쪽에 있는 값을 입력으로 하는 테스트
- 경계선 밖에 있는 값을 입력으로 하는 테스트
- (경계선 안에 있으며) 좀 더 일반적인 값을 입력으로 하는 테스트

❖ **균등분할 테스트와 같이 진행하는 경우가 많음**

❖ 차량 안정성 테스트

테스트 케이스	입력 조건	예상되는 행동
최소값에서	차량 속도 = 0 mph	차는 움직이지 않는다.
최소값 + 1	차량 속도 = 1 mph	차가 뒤집히지 않는다.
최소값 −1	차량 속도 = −1 mph	차의 뒤집힘 방지 소프트웨어 작동하지 않는 상태로 후진한다.
최대값	차량 속도 = 150 mph	차가 뒤집히지 않는다.
최대값 − 1	차량 속도 = 149 mph	차가 뒤집히지 않는다
최대값 + 1	차량 속도 = 151 mph	차량은 지나치게 고속으로 운전하고 있다는 경고 메시지를 준다.
일반적인 경우	차량 속도 = 70 mph차량은 뒤집히지 않는다.	

그림 9.14 자동차 속도만을 고려한 자동차 안정성 소프트웨어에 대한 테스트 스위트

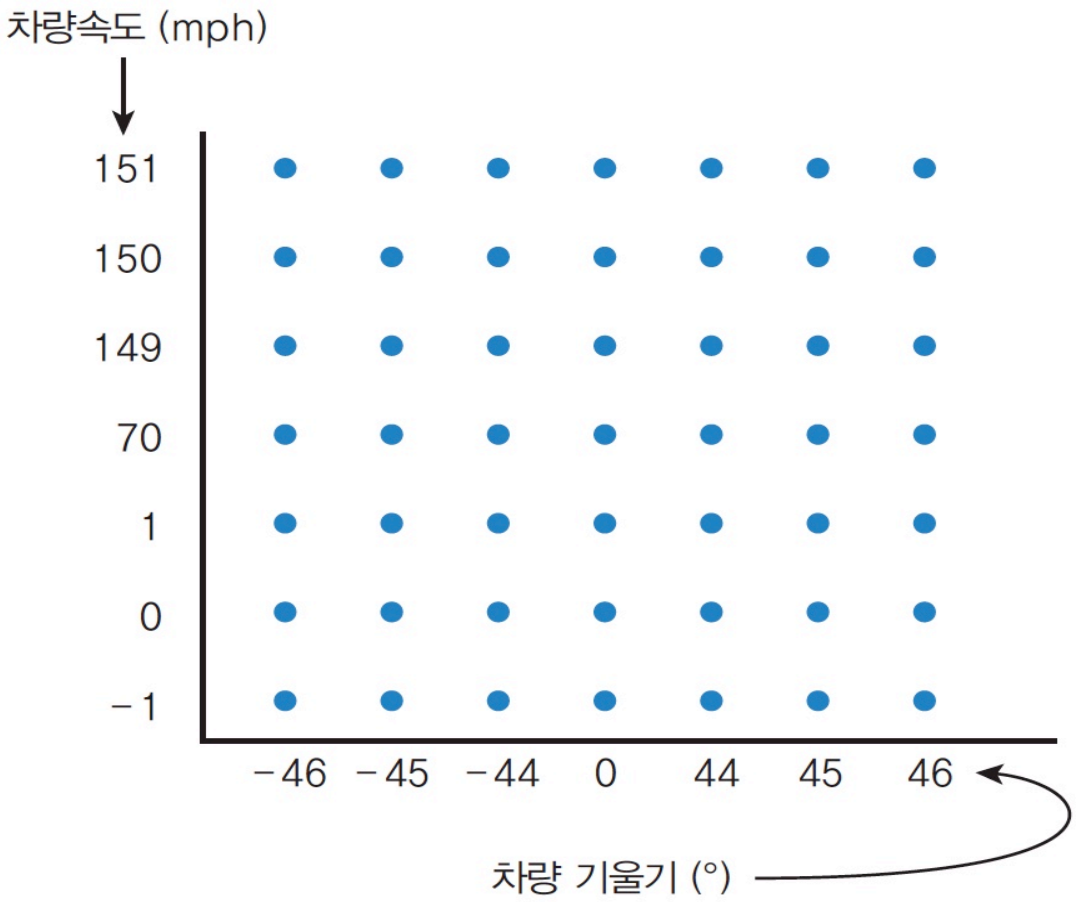

그림 9.15 속도와 기울기를 고려한 경계값 분석

❖ **소프트웨어 테스팅시 사용**

❖ **일상의 다양한 분야에서 사용가능**

- 장난감 제품 조립
- 부하직원 평가
  - 극단 조건 (경계값)에 집중하여 평가를 진행할 수 있다.
- 자동차 구매
  - 자동차 구매시 차량에 대한 원하는 부분의 요구사항 명세서를 만든 후, 각 명세서의 요구사항 별로 다양한 자료를 수집하여 비교 분석할 수 있다.

- 발표방식: ppt 자료 활용 2개 조 각각 10분 발표 (질의응답 포함)

- 문제 1
  - 컴퓨터 시스템들은 매우 고도화된 알고리즘과 기술에 의해 프로그램 되어있기 때문에 컴퓨터 오류라고 불리는 것들은 사실 대부분 사람에 의한 오류이다. 주변의 여러가지 오류 중에서 사람의 오류라고 생각되는 것들과 소프트웨어의 오류라고 생각되는 사례를 조사하여 하나씩 작성하고 왜 그렇게 생각하는지 설명하라.

- 문제 2
  - 소프트웨어 테스팅 방식에는 블랙 박스 테스트와 화이트 박스 테스트가 있다. 이 두 가지 방식의 차이와 두 가지 방식에서 사용하는 기법으로 무엇이 있는지 조사하여 설명하고, 어떤 상황에서 어떤 방식을 사용하는 각각 하나씩 예를 들고 왜 그렇게 생각하는지 설명하시오.