Use Case 기반의 테스트 케이스 도출

신성우*°, 문정봉**, 우종식***
*바산네트워크㈜, **한국게임산업개발원

Test Case Derived from Use Cases

Shin, Sung Woo° , Moon, Jung Bong
BashanNetwork Co. Ltd., Game Infinity
E-mail : swshin@bashannet.com, pongnet@gameinfinity.or.kr, jswoo@gameinfinity.or.kr

요 약

Use Case는 일반적으로 시스템의 기능적 요소를 나타내기 위하여 활용하며, 고객의 요구사항을 반영하여 작성을 한다. 그러므로 Use Case 규격대로 시스템이 구현되었는가를 검증하는 것은 고객이 원하는 시스템이 구현되었는가를 검증하는 것이라 할 수 있다. 그러므로 Use Case를 기반으로 테스트를 수행 하기 위한 테스트 케이스 작성이 필요하다. 본 논문에서는 Use Case를 이용하여 테스트 케이스를 도출하는 과정을 제안하고, 이에 의하여 확보되는 테스트 커버리지의 의미를 설명한다.

1. 서론

최근에는 UML을 기반으로 한 시스템 설계, 분석이 점차 보편화되고 있다. UML의 여러 다이어그램 중 Use Case는 시스템의 기능적인 면을 나타내는데 활용되고 있다.[1] 특히 Use Case는 쉽게 활용할 수 있으며, 전문가가 아니어도 어렵지 않게 이해할 수 있기 때문에 더욱더 대중적으로 활용이 되는 추세이다. 우리는 다음과 같은 목적을 위해 Use Case를 활용하고 있다.[2]

- 여러 이해당사자간의 의사소통 향상
- 시스템요구사항을 이해하는데 도움을 줌
- 세부적인 절차(How)를 몰라도 핵심적인 기능(What)을 쉽게 파악 가능

그런데 Use Case는 시스템의 기능적인 요구사항을 나타내기 때문에 시스템 개발 후 Use Case대로 시스템이 구현되었는가를 검증하는 것은 고객의 기능적 요구사항대로 시스템이 개발되었는가를 검증하는 것이라 할 수 있다. 다시 말하여 Use Case를 기반으로 시스템을 테스트 해 볼 필요가 있으며, 테스트를 수행하기 위한 테스트 케이스의 작성이 필요하다. 본 고에서는 Use Case를 기반으로 하는 테스트 케이스[3] 도출 과정과 이에 의하여 확보되는 테스트 커버리지의 의미를 설명하겠다..

2. Use Case와 테스트 관계

그림 1을 참조하여 Use Case의 구성을 살펴보면 Use Case는 시스템 외부에서 활동을 하는 액터(Actor), 둥근 원 형태의 Use Case, 액터와 Use Case

를 연결하는 관계(Relationship) 및 설명부분(Description)으로 구성되어있다.

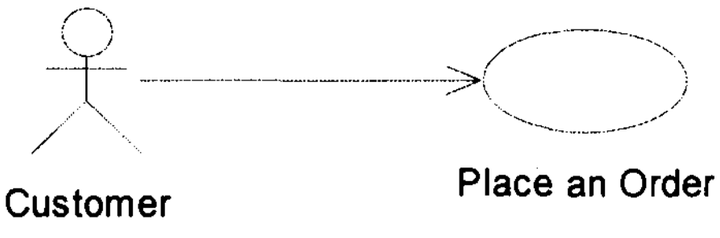


그림 1 Use Case 다이어그램

Use Case를 이용하여 시스템을 개발하는 경우, 일차적으로 고객의 요구사항을 바탕으로 시스템의 기능적 요구사항을 나타내는 Use Case를 작성하게 된다.[4] 그 후 이것을 활용하여 시스템에서 사용할 객체(Object)을 식별하게 된다. 그 후 각 Use Case에 대한 시나리오를 작성해서 이 것을 바탕으로 상호작용(Interaction) 다이어그램을 작성하게 된다. 결국에는 시스템에서 사용할 클래스 다이어그램을 도출하게 된다. 이러한 일련의 소프트웨어 개발 절차와 소프트웨어 테스트의 관계는 그림 2와 같다.

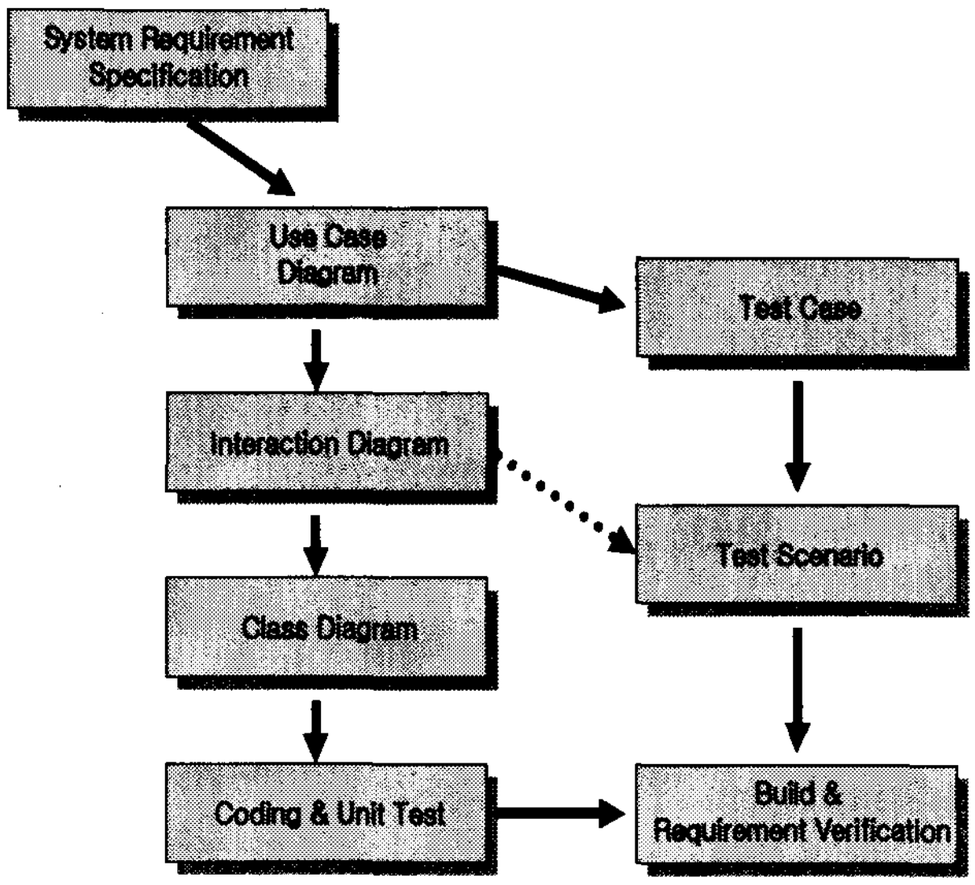


그림 2 소프트웨어 개발 절차와 테스트 절차의 관계

그림 2를 살펴보면 Use Case를 바탕으로 테스트 케이스를 도출하였다. 테스트 시나리오는 Use Case의 이벤트 흐름(event flow)을 바탕으로 작성한 시스템의 시나리오인 상호작용(Interaction) 다이어그램을 활용하여 작성하게 된다. 다음단계로는 코딩 및 유닛 테스트 과정을 거치게된다. 최종적으로는 완성된 시스템을 테스트 시나리오를 활용하여 소프트웨어 요구사항을 만족시키는가를 검증하게 된다. 이와 같이 Use Case를 기반으로 소프트웨어를 개발하고 테스트 케이스, 시나리오를 작성하게 되면 상호간의 추적성(Traceability)를 보장[5]하면서 시스템 요구사항을 검증할 수 있게 된다.

다음 절에서는 이 과정을 보다 자세히 이해하기 위한 Use Case를 설명한다.

3. Use Case

Use Case는 직관적으로 이해할 수 있는 개념을 나타내는 부분과 세부적인 설명을 기술하는 텍스트 부분으로 구분할 수 있다. 이중 텍스트 부분은 다음과 같은 요소로 구성이 된다[6].

- 이름(Name)
- 간단한 Use Case 설명
- 소프트웨어 요구사항 설명(SRS supports)
- 선,후 조건 기술
- 이벤트 흐름(Event flow)

위의 여러 구성요소 중 테스트에 가장 중요한 부분은 이벤트 흐름(Event flow)이다. 이벤트 흐름이란 액터가 시스템과 상호작용을 하는 과정을 기술한 것이다. 시스템을 블랙박스처럼 생각한다면, 이벤트 흐름을 진행하다 보면 여러 가지 예외사항(Exception)에 의하여 다양한 흐름이 발생할 수 있음을 알 수 있다. 다음은 일반적인 Login시에 발생할 수 있는 이벤트 흐름이다.

이벤트 흐름: Login

- 사용자가 Login을 위해 필요한 ID 입력창을 선택하면 Use Case가 시작된다.
- 사용자가 ID를 입력한다.

- 사용자가 Password를 입력한다.
- 사용자가 Login버튼을 누른다.
- 시스템은 사용자 정보가 올바른가 검증한다. [E1, E2]
- 사용자가 시스템에 Login한다.[E3]

예외사항(Exception):

E1 : 사용자가 잘못된 ID를 입력한다.
시스템은 사용자에게 잘못된 ID를 입력하였다고 알려준다.
초기화면 버튼을 누르면 초기화면으로 되돌아간다.
Use Case가 종료된다.

E2 : .....

시나리오는 Use Case의 이벤트 흐름을 따라가는 하나의 과정을 말한다. 이 때 아무런 예외사항도 거치지 않고 종결되는 Use Case를 해피 시나리오(Happy scenario)라고 한다. 일반적으로는 예외사항의 수에 따라 많은 수의 시나리오가 작성된다. 그림 3은 하나의 Use Case에서 여러 개의 시나리오가 발생할 수 있음을 보여준다.

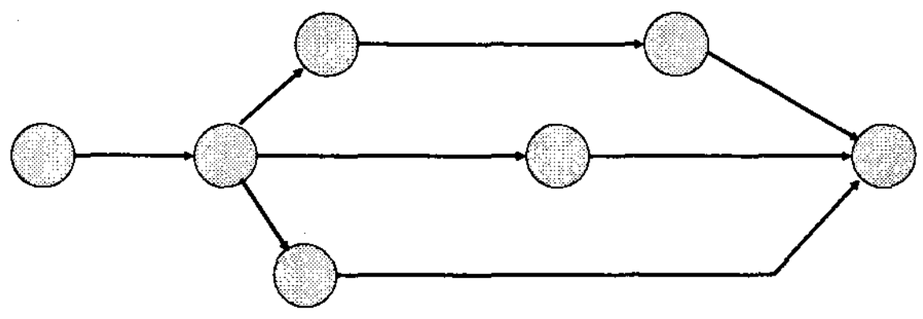


그림 3 : 이벤트 흐름의 다양한 시나리오

Use Case와 테스트의 관계를 설명하기 위하여 다음의 문제(Problem statements)를 사용하도록 하겠다.

인터넷 홈쇼핑사이트를 제작하려고 한다. 이 사이트에서 고객은 원하는 물건을 주문을 할 수가 있다. 일단 주문을 하게 되면 주문 진행상태를 확인할 수 있으며, 원할 경우 주문한 물건을 취소할 수도 있다.

물건 주문 시에는 고객의 신용카드 정보를 확인한다.
만약 고객이 선택한 물건을 현재 가지고 있지 않는 경우에는 백오더(back-order)를 원하는 가를 확인한다.

위의 문제(Problem statement)를 분석하면 그림 4와 같은 Use Case 다이어그램이 나온다.

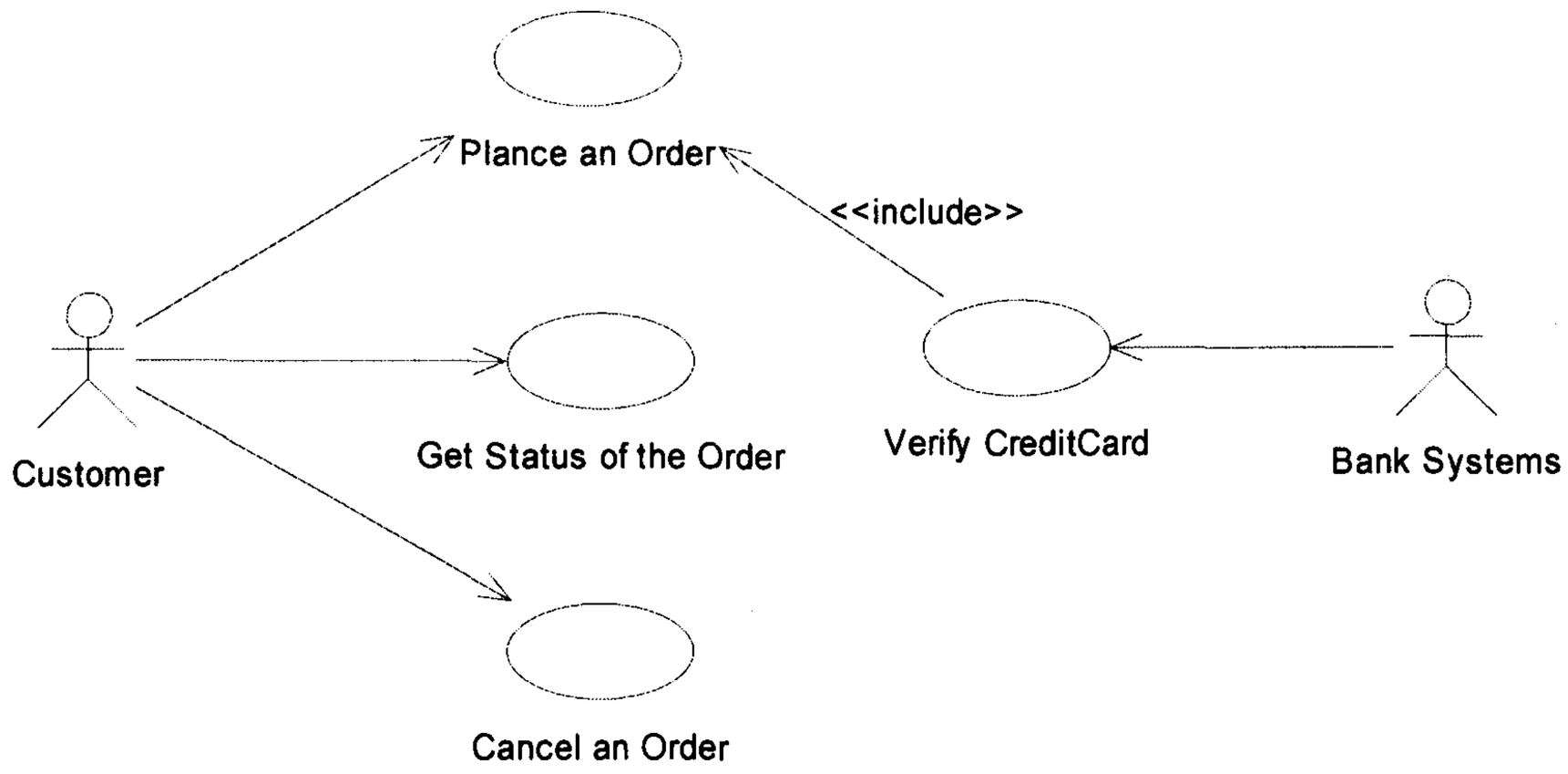
이 Use Case의 설명은 다음과 같다.

- Place an Order : 고객이 원하는 물건을 선택한 후 주문을 한다.
- Get Status of the Order : 고객이 자신이 주문한 물건의 배송 상태를 확인한다.
- Cancel an Order : 고객이 자신이 주문한 물건을 취소한다.
- Verify Creditcard : 은행시스템에서 고객의 신용카드가 정상적인가를 확인한다.

위의 여러 Use Case중에서 Place an Order Use Case의 경우에는 다음과 같은 이벤트 흐름이 발생할 수 있다.

Place an Order 이벤트 흐름

- 고객이 원하는 제품을 선택 후 구매 버튼을 누르면 Use Case가 시작된다.
- 고객이 구매 제품의 개수를 입력한다.[E1]
- 시스템은 총 비용을 알려준다.
- 고객이 확인 버튼을 누른다.[E2]
- 시스템은 신용카드 정보를 묻는 화면을 보여준다.
- 고객은 신용카드 정보를 입력한다.
- 은행시스템은 신용카드 정보를 확인한다.[E3] [E4]
- 시스템은 구매가 성공적으로 끝났음을 알려준다.
- Use Case가 종료된다.



위의 이벤트 흐름을 활용하면 Place an Order Use Case에 대하여 다음과 같이 이벤트 흐름을 작성해 볼 수 있다.

1. 고객이 원하는 제품을 선택 후 구매 버튼을 누르면 Use Case가 시작된다.
2. 고객이 구매 제품의 개수를 입력한다.
   예외사항 1: 고객이 주문한 제품의 개수보다 쇼핑몰에 있는 제품의 수가 적은 경우
   - 시스템은 제품의 수량이 부족함을 고객에게 알려준다.
   - 벡오더를 할 것인가를 물어본다.
   - 시스템은 제품의 주문이 제고보다 많음을 관리자에게 알려준다.
   - 제품 수량 입력화면으로 돌아간다.
3. 시스템은 총 비용을 알려준다.
4. 고객이 확인버튼을 누른다.
   예외사항 2: 고객이 취소 버튼을 누르는 경우
   - 시스템은 구매에 관계된 데이터를 초기화한 후 제품 카다로그 화면으로 돌아간다.
   - Use Case 종료
5. 시스템은 신용카드 정보를 묻는 화면을 보여준다.
6. 고객은 신용카드 정보를 입력한다.
7. 은행시스템은 신용카드 정보를 확인한다.
   예외사항 3: 고객이 신용카드 정보를 잘못 입력한 경우
   - 은행시스템은 현 상태에 대한 로그를 남긴다.
   - 은행시스템은 신용카드 정보가 잘못 되었음을 고객에게 알려준다.
   - 신용카드정보 입력 화면으로 돌아간다.

   예외사항 4: 도단 신용카드인 경우
   - 은행시스템은 현 상태에 대한 로그를 남긴다.
   - 은행시스템은 사용자의 IP를 확인한다.
   - 은행시스템은 현 사용자의 정보를 경비시스템에 전달한다.
   - 은행시스템은 신용카드 정보가 잘못 되었음을 고객에게 알려준다.
   - 신용카드정보 입력 화면으로 돌아간다.
8. 시스템은 구매가 성공적으로 끝났음을 알려준다.
9. Use Case가 종료된다.

다음 절에서는 위의 이벤트 흐름을 검증하기 위한 테스트 케이스 작성 사례를 설명한다.

4. 테스트 케이스 작성

Place an Order에 대한 테스트 케이스

Test 1 : 해피 시나리오 고객이 성공적으로 주문을 마치는 경우

- 고객이 주문 버튼을 누름
- 고객이 입력한 제품 수량이 적합한가 검증
- 고객이 입력한 신용카드 정보가 올바른가 검증
- 고객에게 구매 완료 정보를 제공하는가 검증

Test 2 : 고객이 주문한 제품의 개수보다 쇼핑몰에 있는 제품의 수가 적은 경우

- 시스템이 제품의 수량이 부족함을 고객에게 알려주는 가를 검증
- 벡오더를 할 것인가를 물어보는가를 검증
- 시스템이 제품의 주문이 제고보다 많음을 관리자에게 알려주는가를 검증
- 제품 수량 입력화면으로 돌아가는 가를 검증

Test 3 : 고객이 취소 버튼을 누르는 경우

- 구매에 관계된 데이터가 초기화 되는가를 검증
- 제품 카드로그 화면으로 돌아가는가 검증

Test 4 : 고객이 신용카드 정보를 잘못 입력하는 경우

- 은행시스템이 현 상태에 대한 로그가 남는지 검증
- 은행시스템이 신용카드 정보가 잘못 되었음을 고객에게 정상적으로 알려주는가를 검증
- 신용카드 정보 입력 화면으로 돌아가는 가를 검증

Test 5 : 도난 신용카드로 주문을 하는 경우

- 은행시스템이 현 상태에 대한 로그를 남기는가를 검증
- 은행시스템이 사용자의 IP를 확인하는가 검증
- 은행시스템이 현 사용자의 정보를 경비시스템에 전달하는가를 검증
- 은행시스템이 신용카드 정보가 잘못 되었음을 고객에게 정상적으로 알려주는가를 검증
- 신용카드 정보 입력 화면으로 돌아가는 가를 검증

이와 같은 방법을 사용하면 쉽고 효과적으로 테스트 케이스를 작성이 가능하다. 다만 이 방법은 Use Case가 시스템의 기능적인 면을 나타내기 때문에 기능적인 요구사항에 대한 테스트 케이스만 개발이 가능하다는 단점이 있다. 시스템의 성능, 신뢰성 등을 테스트 하기 위해서는 별도의 방법을 사용하여 테스트 케이스를 작성해야만 한다. 만약 Use Case의 수가 많고 복잡한 예외조건이 사용되는 시스템의 경우에는 적절하게 우선순위를 부여하여 실제로 수행이 가능한 수준의 테스트 케이스를 작성해야 하는 경우도 있다.

5. 결론

Use Case를 기반으로 테스트 케이스를 작성하게 되면 크게 세가지 이점이 있다. 첫째로는 Use Case대로 정말 시스템이 개발되었는가를 검증해 볼 수 있다. 두번째로는 매우 빠른 시간 내에 테스트 케이스를 작성하여 테스트를 수행해 볼 수 있기 때문에 여분의 시간에 좀더 다양한 테스트를 수행해볼 수 있다. 뿐만 아니라 개발 초기부터 테스트 케이스를 작성 할 수 있기 때문에 시스템 개발 완료단계에서 쫓기면서 테스트를 수행할 필요가 없다. 세번째로는 Use Case 개수에 의하여 테스트 케이스의 개수가 영향을 받기 때문에 우리는 총 테스트 해야 할 대상 중에서 실제로 테스트를 수행한 대상이 얼마만큼 된다는 것을 쉽게 파악할 수 있다는 점이다. 이러한 테스트 커버리지의 확보는 수행한 테스트의 완성도가 얼마만큼되는가를 파악 할 수 있게 해주는 것이며 이를 통해 개발한 시스템의 품질에 대하여 보다 확신을 가질 수 있게 된다.

[참고문헌]

[1] The Unified Modeling Language User Guide: Grady Booch and James Rumbaugh and Ivar Jacobson, Addison Wesley, 2000

[2] Applying Use Cases, Second Edition-A Practical Guide: Geri Schnider and Jason P. Winters, Addision Wesley, 2001

[3] Effective Methods for Software Testing: Perry, Wiley, 2000

[4] Software Requirements-Best Practice: Karl Wiegers, Microsoft, 1999

[5] Requirement Management with Use Case, Rational University Course wares, 2001

[6] Object Oriented Analysis and Design, Rational University Course wares, 2000