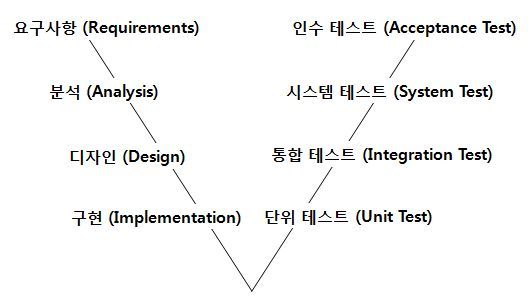
**프로젝트의 개요**

1. **프로젝트의 배경**

노출되지 않은, 숨어있는 결함을 찾기 위해 소프트웨어를 작동시키는 일련의 절차를 테스트라고 하며, 이 활동은 소프트웨어 개발 라이프사이클의 중요한 과정 중 하나임을 알고 있을 것이다.



이처럼 테스트는 각 과정에서 거쳐야 할 작업이지만, 현실에서는 그 중요성을 간과하고 개발일정의 지연 혹은 개발비용 등의 이유로 테스트 기간을 짧게 잡거나 아예 없애버리는 경우도 있다. 또한 테스트 기간을 충분히 잡았다고 하더라도 테스트 계획이 제대로 짜여져 있지 않거나 테스트에 대한 잘못된 접근법이나 방법을 사용했다면 이것은 잠재적인 치명적 결함이나 버그로 이어질 수 있기 때문에 테스트는 개발보다 더 중요하다고 생각한다.

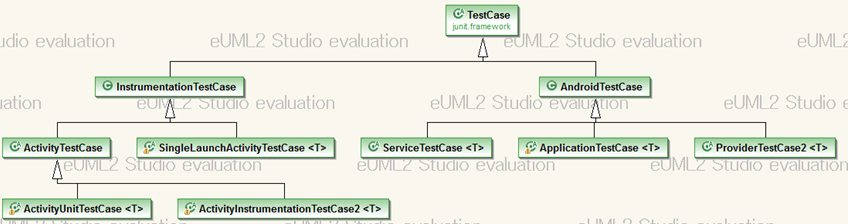
테스트에 대한 잘못된 접근. 예를 들어 DB관련 제품 설계 혹은 유지 및 보수의 과정에서 실제DB에 접근하여 제어하는 방법은 정말 큰 재앙이 아닐 수가 없다. 이러한 상황에서 테스터들에게 가상의 데이터베이스를 통한 테스트 환경을 제공하고 테스트 케이스 및 테스트 결과에 대한 산출물을 자동화 한다면 테스트 시간과 비용을 절감 할 수 있을 것이다.

1. **프로젝트의 목적**

이러한 테스트 환경을 제공하기 위해서 본 프로젝트는 사용자에게 테스트를 위한 가상의 데이터베이스를 제공하여 테스트에 제한적이지 않은 환경을 제공하고 사용자가 다루기 쉬운 인터페이스를 제공할 수 있어야 한다.

이와 더불어 테스트케이스와 같은 문서, 테스트에 대한 결과의 이력 등을 자동화하여 테스트 시간과 비용의 효율을 극대화하고 나아가 소프트웨어의 품질 향상을 유도 할 수 있는 프로젝트 완성을 목표로 한다.

1. **프로젝트의 필요성**

****

Android의 Test Class는 위 그림과 같이 총 9개로 구성되어 있다. 각각 특정 Component를 테스트 하기 위하여 특화되어 있다. 위 그림을 자세히 보면 Test Class들이 두 가지 유형으로 구분되어 있다는 것을 발견할 수 있을 것이다.

하나는 Instrumentation과 Test Case Class를 상속받은 Class이고 하나는 그냥 Test Case만 상속 받은 Class이다.

Instrumentation Test 기법의 장점은 사람이 일일이 귀찮게 여러 번 테스트 하는 것을 흉내 낼 수 있어 자동화로 인한 이점이 큰 반면에 만일 계속 그 형태가 바뀌는 상황이라면 유지하기가 어려워 지고 한번 쓰고 버리기 쉽게 되어버린다. 반면, JUnit 기법은 Instrumentation기법과 정반대의 장단점을 가지고 있다. JUnit기법의 가장 큰 장점은 유지보수가 비교적 쉽고, 테스트 수행 자동화가 쉽다는 데에 있다. 소스코드는 수없이 변할지 몰라도 method의 signature은 쉽게 변하지 않는다. 즉 한번 만들어 놓으면 재사용이 가능하고 테스트 수행을 자동화 해놓기가 매우 쉽다. 다만 Instrumentation테스트 기법과는 다르게 테스트 범위가 매우 한정적이며, 시간과 비용이 많이 들게 된다.

이러한 Instrumentation Test기법과 JUnit Test기법의 단점을 버리고 장점을 갖는 테스트 환경을 제공하고자 함이다.