Test Execution

Introduction

- Designing test cases requires sufficient knowledge of problem domain
- Designing is creative and Executing test cases should be mechanical
- Manual Test execution requires
  - More time
  - Employee hours are wasted
  - More cost
  - Human eye is slow, expensive & unreliable for judging test outcomes
- Automation
  - Can be run at night
  - A large suite of test data may be generated automatically from compact and abstract set of test case specifications

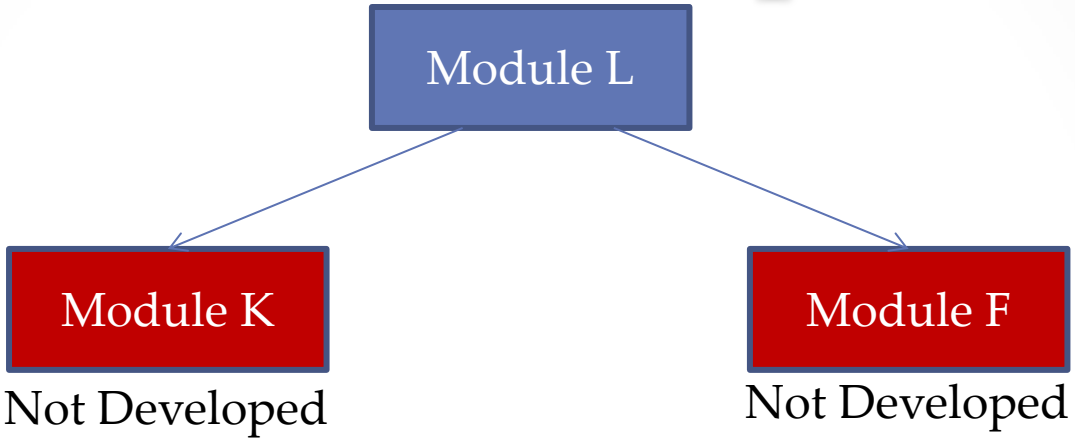
Scaffolding

- Most of the time only portion of the full system is available for testing
- Since various modules are linked to each other we need to add some extra code to give a feel of system execution
- Code developed to facilitate testing is called Scaffolding (Temporary structure erected around a building during construction)

Scaffolding

- Scaffolding includes:
  - Test stub: substituting for functionality called
  - Test harness: substituting for parts of the deployment environment
  - Test driver: substituting for a main or calling function
- Top down testing
  - We first create the root module after that start creating each module which is called by the root.
- Bottom up testing
  - We first develop the leaf nodes.
- Why scaffolding?
  - Controllability to execute test cases and observability to judge the outcomes of test execution
    - Eg: interactive program that is normally driven through Gui. Small driver programs Independent of Gui can drive each module through large test suites in a short time.

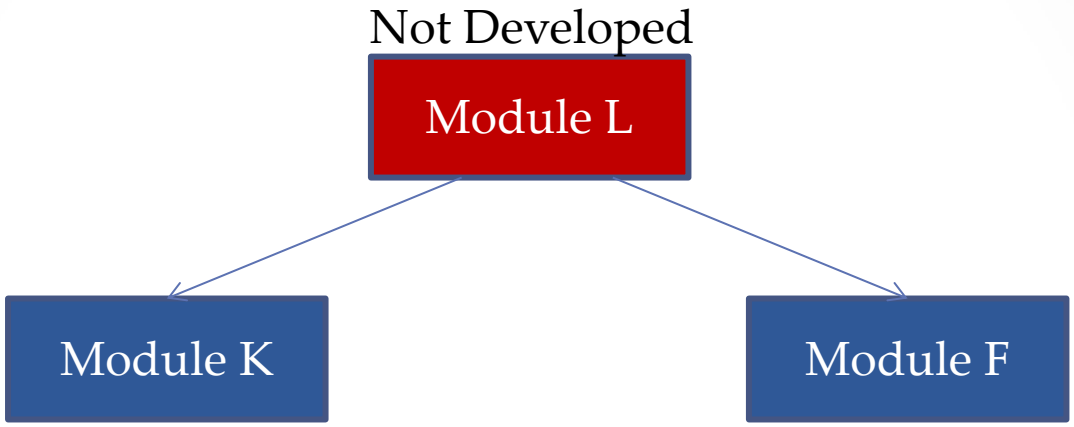
Stub Example




```
void functionForTest(parms ... )
{
        …
        int p = price(param1);
}

void price(int p ) // this is a stub
{
        return 10; //value doesn't matter
}
```

Driver Example




```
void price(int p )
{
        //complex calculation
}
void driverFunction(parms ... )
{
        …
        int p = price(param1);
        print(p)
}
```