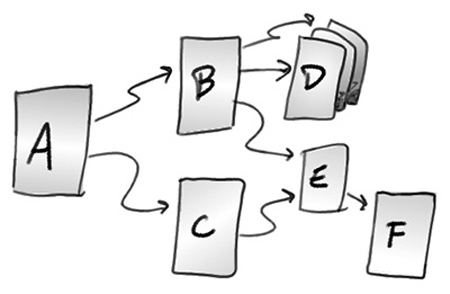
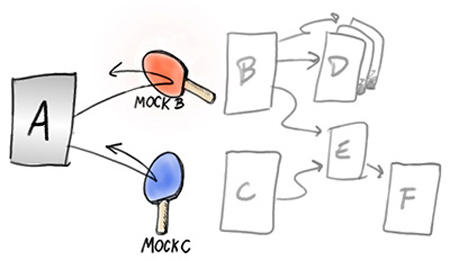
The mock object approach to test-driven development

* As embedded applications grow in size they are often broken into subsections that we call modules
* When you get to a point where you need a function, or a module that doesn’t exist, you simply mock it out and continue with the current module.



Mock:

A mock is a stand-in for a real module. It doesn’t contain any real functionality but rather imitates a module’s interface .If the function being called has a return value, the mock for that function will also return a value as specified in the test. Additionally, mocks have a special feature that keeps track of the number of times each function is called, and the order of those calls.



Mocking framework:

A mocking framework is a tool that generates mocks for you. For example:

* cMock for C : To generate a mock, all it needs is a header file. CMock parses the function declarations in the header file and generates mock functions based on the prototypes
* googleMock for c++
* cppUTest for c++
* creedling

Testing frameworks and mocking frameworks are generally used together within a build automation environment.

|  |  |  |  |
| --- | --- | --- | --- |
| // test\_LedControl.h #include “LedControl.h” #include “mock\_Gpio.h” void test\_LedControl\_TurnLedOn\_should\_set\_GPIO\_pin\_1\_when\_turning\_on\_ the\_red\_LED() {    // Setup expected call chain    GPIO\_SetPin\_Expect(1U);     // Call the function under test    LedControl\_TurnLedOn(LED\_RED); }  Listing 1 - Unit Test | // Gpio.h void GPIO\_SetPin(uint8\_t pin\_num);  cMock will create a mock\_Gpio.h corresponding to it  Listing 2 - GPIO Module Header File | // LedControl.h  #define RED\_LED\_PIN 1 #define BLUE\_LED\_PIN 2  typedef enum {    LED\_RED,    LED\_BLUE } LED\_T;  void LedControl\_TurnLedOn(LED\_T led);  Listing 3 - LedControl Header File | // LedControl.c  #include “LedControl.h” #include “Gpio.h” void LedControl\_TurnLedOn(LED\_T led) {    if(LED\_RED == led)    {       GPIO\_SetPin(RED\_LED\_PIN);    } }  Listing 4 - LedControl Source File |

* In many cases, a given module will make calls into one or more neighboring modules. Those modules, in turn, make calls into other modules and the chain continues all the way down to the lowest-level hardware interactions.