Functional integration test planning

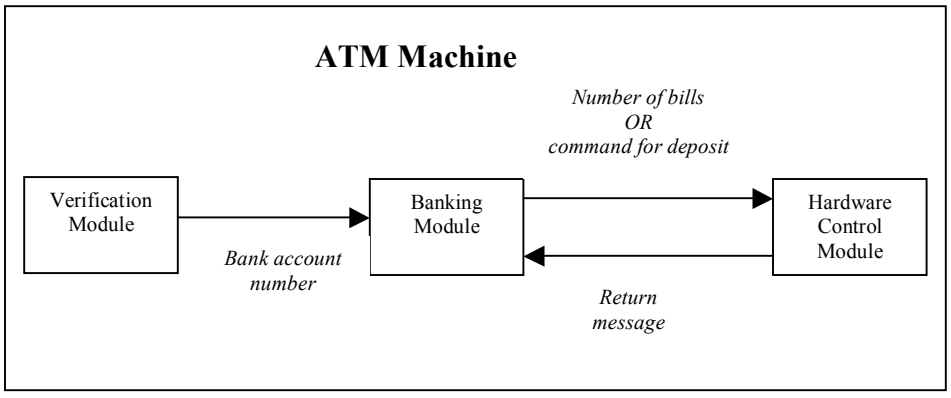
By Tim Van Tongeren <tim@testgeek.com>

Integration testing is the process of validating proper functionality between two or more components. Some shops use different definitions of a "component", so first one should determine the type of integration being performed. The most common use of the concept of integration testing is directly after unit testing. A unit will be developed, it will be tested by itself during unit test, and then it will be integrated with surrounding units in the program. Integration testing could also be used to describe the validation of multiple programs in an application, multiple applications in a company, or multiple companies in a network.

With an ATM machine, there may be several units that verify user data: bank account verification, PIN verification, and bank account status verification. All of these individual units would be integrated into the Verification module. The Verification module would be integrated with a Banking module and a driver for the mechanical pieces, into the entire ATM application. After this testing was complete, integration testing could occur between the ATM machines and the bank's ATM application server. If the bank acquires another bank, the corporate ATMs would have to be integrated into a new network. Each cycle of integration adds another layer until a comprehensive test can be performed on the entire system.

Design

Now let's see how we can build test cases for functional integration testing. (We will not include non-functional requirements concerning performance, usability, or security in this example.) Suppose each of our ATM machines contains a verification module, a banking module and a hardware control module. For this example, we will not cover every process inside the ATM and we won't even touch the bank, just to keep it simple. Let's look at our hypothetical design.



- The verification module authenticates the user based on card number, PIN number and bank account status. Upon successful authentication, it will pass the bank account number to the banking module.

- The banking module receives the bank account number and prompts the user for allowable actions. The user can check account balances, request a withdrawal, or make a deposit. If a withdrawal is selected and money needs to be delivered to the user, the banking module sends the hardware control module a number, which indicates the number of $20 bills to deliver. If a deposit is selected, the banking module sends the hardware control module the command to accept the deposit.

- For a withdrawal, the hardware control module receives the number from the banking module and attempts to deliver the required number of $20 bills. If it can deliver all the bills, it returns a success message to the banking module. If it cannot deliver all the bills, it returns an error to the banking module.

- For a deposit, the hardware control module receives the command to accept the deposit. Funds from deposits are not immediately available for withdrawal. Deposits will be verified by a cashier at the bank for validity. Because of this, the ATM will not behave differently for failed deposits.

- The banking module receives the return message from the hardware control module. If the message indicates a success, the banking module records the debit transaction. If the return message indicates a failure, the banking module records a failed attempted withdrawal.

Test Planning

In the verification module, we don't need to test each scenario of failure (invalid bank account, unreadable card, wrong PIN for bank account number, or invalid bank account statuses such as expired card, inactive member or deleted member.) These would each be tested during unit and system testing of the verification module. Remember, the point of integration testing is to verify proper functionality between components, not to retest every combination of lower-level functionality. So we will want to test each unique message/action between the modules. Since the verification module only sends a message to the banking module if there is a successful authentication, we would only test a single failed authentication as a negative test, in order to verify that the verification module does not send a message to the banking module.

Inside the banking module, there is no integration with the hardware control module during the following scenarios: balance check, attempt to withdraw more money than the account contains and cancelled transaction. However, we don't need to test all of these scenarios, so we would only test one of them as a negative test. Positive tests would be run on the withdrawal and deposit processes. There are two messages the banking module can pass to the hardware control module: deposit and withdrawal. There are two messages the hardware control module can send back to the banking module: withdrawal success and withdrawal failure.

Test Cases

Test Case #1: Deposit - Use a valid card with associated PIN to login to the ATM. Verify that the verification module passes bank account number to the banking module. Select deposit. Verify that the banking module passes the command to accept the deposit to the hardware control module.

Test Case #2: Withdrawal success - Use a valid card with associated PIN for an account with a $100 balance to login to the ATM. Verify that the verification module passes bank account to the banking module. Select withdrawal of $40. Verify that the banking module passes the command to deliver $40 to the user. Verify that the ATM delivers $40. Verify that the banking module records a successful withdrawal transaction.

Test Case #3: Withdrawal failed - Use a valid card with associated PIN for an account with a $20 balance to login to the ATM. Verify that the verification module passes bank account to the banking module. Select withdrawal of $100. Verify that the banking module passes the command to deliver $100 to the user. Verify that the ATM delivers no money. Verify that the banking module records a failed withdrawal transaction.

Test Case #4: Negative test between verification module and banking module - Use a valid card with an incorrect PIN to attempt to login to the ATM. Verify that the verification module does not pass the bank account number to the banking module.

Test Case #5: Negative test between banking module and hardware control module - Use a valid card with associated PIN to login to the ATM. Verify that the verification module passes bank account to the banking module. Select the option to check account balance. Verify that the banking module does not pass anything to the hardware control module.

Conclusion

The example we looked at is fairly typical of a functional integration test. It demonstrates several components that pass messages between each other. It is at each junction between components that integration testing occurs. In our case, the components were software modules inside an application. In your case, it may be at the module, application, system or corporate level. Once you determine the level at which you need to test, the next step is to understand each component at that level. Then you can determine each component interaction. With this information, you should be able to put together a comprehensive set of test cases.