[ATMSD-2] Withdrawal function test for ATM System

Created: 22/Aug/16 2:17 PM - Updated: 23/Aug/16 3:10 PM - Due: 26/Aug/16

Status:	To Do
Project:	ATM System Design
Component/s:	ATM Application, Bank Application

Type:	Test	**Priority:**	Medium
Reporter:	Levente Szabo	**Assignee:**	Owen Klyed
Resolution:	Unresolved		
Labels:	ATM, interactions		

Test Details

Estimated execution time (h):	6.5

Approvals

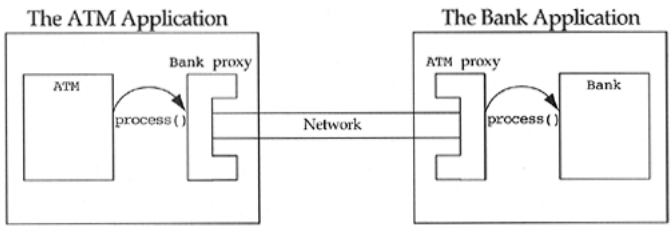
Approved by:	Casey Ford, Dalia Lens, Robert Mongose
Final approval date:	01/Sep/16

Execution Requirements

Assets required for execution:	Test-ready ATM system ATM display Keyboard Receipt printer Cash dispenser
Security clearance required for execution:	C2

Description

Test case of basic function "Withdrawal" to verify that the implementation is basically correct.



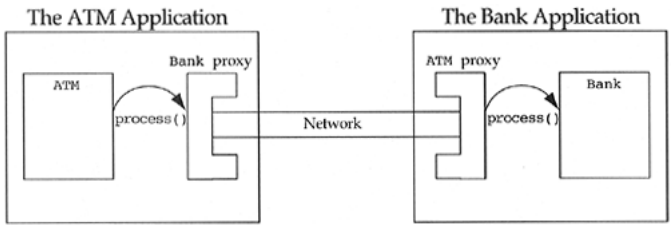
At this point the **ATM** needs to send a message to the **Bank** object, asking it to process a transaction (passing the **Withdraw transaction** object as an explicit argument). The **ATM** object lives in one address space (the **ATM** application) but the Bank lives in a different address space (the **Bank** application). We will employ proxies to make the **ATM** and the Bank objects viewed in the same address space.

The **ATM** cannot send a direct message to a **Bank**, so it sends a message to a **Bank** proxy that lives in the **ATM's** address space (see attachement). This proxy packs up the request and transaction object and ships it across the network to an **ATM** proxy that lives in the **Bank's** address space. The **ATM** proxy unpacks the request, reconstitutes the transaction object, and sends the process message to the real Bank object.

The real **ATM** and **Bank** are completely unaware that they are really talking to proxies. This allows us to ignore the distributed facet of a distributed application during high-level design, leaving the gory details to low-level design proxy classes.

Test Step	Test Data	Expected Result
1. Initiate a connection to **BANK**	Use TCP/IP over Ethernet.	Connection to **Bank** has been established.
2. Insert a readable card		System asks for entry of **PIN**
3. Enter **PIN**		System displays transaction types
4. Choose Withdrawal transaction		System displays account types
5. Choose checking account		System displays possible withdrawal amounts
6. Choose amount that • the ATM currently has and • is not greater than the amount available		Messsage is shown: > Verifying account balance
7. Cancel transaction during verification		Messsage is shown: > Withdrawal cancelled

Attachments



ATM-sysdesign.gif (21 kB)

Links

Bugs detected

detects	[~~ATMSD-10~~]	Bank proxy doesn't provide available accounts	Done
detects	[~~ATMSD-11~~]	No transaction options when correct PIN entered	Done

Requirements verified

verifies	[ATMSD-3]	Connection can be initiated while in idle state	Defined
verifies	[ATMSD-4]	System asks for PIN when readable card is inserted	Defined
verifies	[ATMSD-5]	System verifies PIN number	Defined
verifies	[ATMSD-6]	When corrent PIN is entered, transactions menu is shown	Defined
verifies	[ATMSD-7]	When transactions is selected, Bank sends a list of available accounts	Defined
verifies	[ATMSD-8]	System keeps track of money on hand	Defined
verifies	[ATMSD-9]	Transaction can be cancelled at any state	Defined

Comments

Casey Ford added a comment - 23/Aug/16 3:06 PM

However I think I get the idea behind this design decision, but wouldn't it be a better alternative to have the `Bank` simply tell its transaction object to process itself, handing it the whole list of accounts?

Owen Klyed added a comment - 23/Aug/16 3:10 PM

Yes, I can see point Casey Ford. In this way, the particular process method, which runs for a given transaction type, can be responsible for determining the selection of account object(s). It also allows us to keep related data and behavior closer together by eliminating the removal of the account number from the transaction object.