Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

Example (Intro to ANSA)

Yigal Hoffner

Abstract

The business problem addressed is...

The technical problem created by that business problem is ...

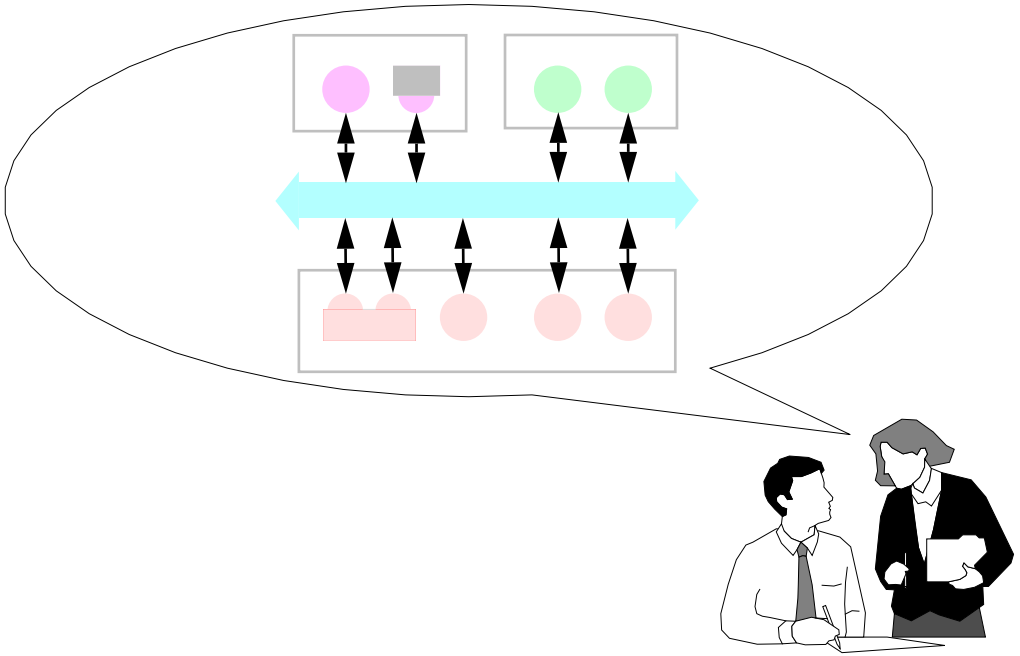
The solution being offered is....

APM.1652.00.01 **Draft** 2nd November 1995

Briefing Note

Distribution:
Supersedes:
Superseded by:

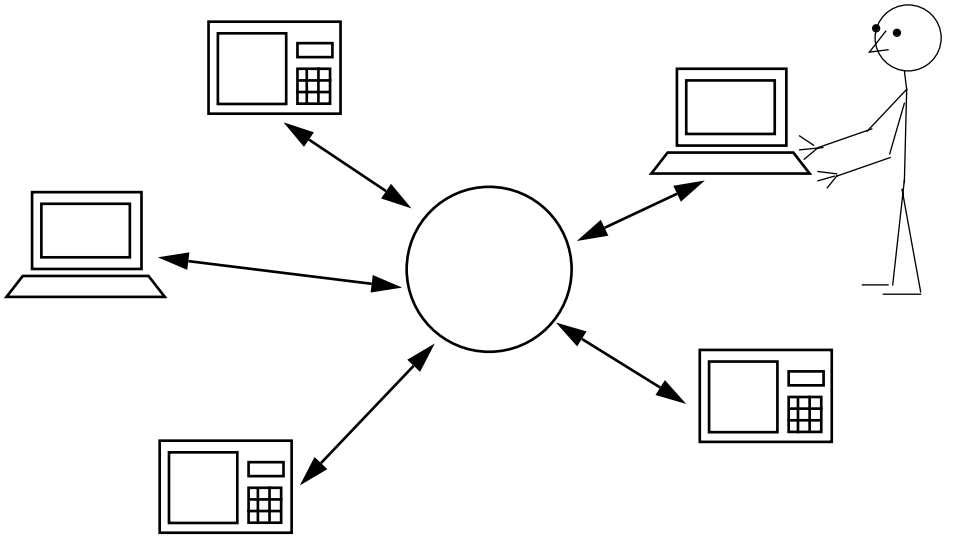
A Distributed Application



In this session

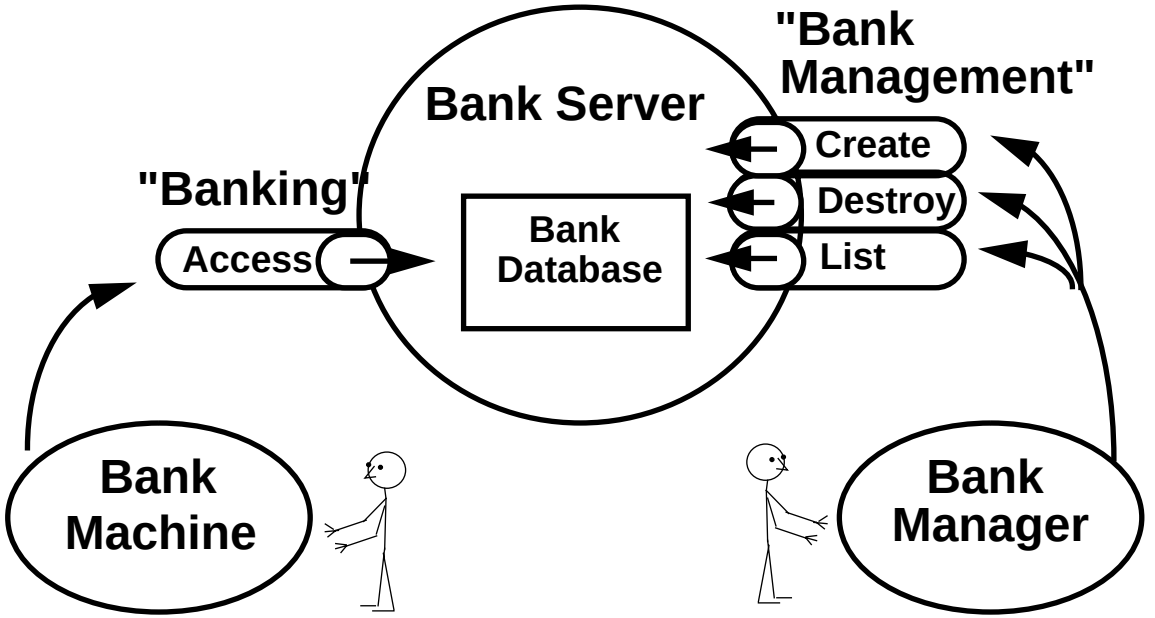
- ***Look at an example application***
  - **used in the hands-on course**
- ***Examine how the ideas and technology described earlier can be brought together to solve real problems***

Simple Bank Example



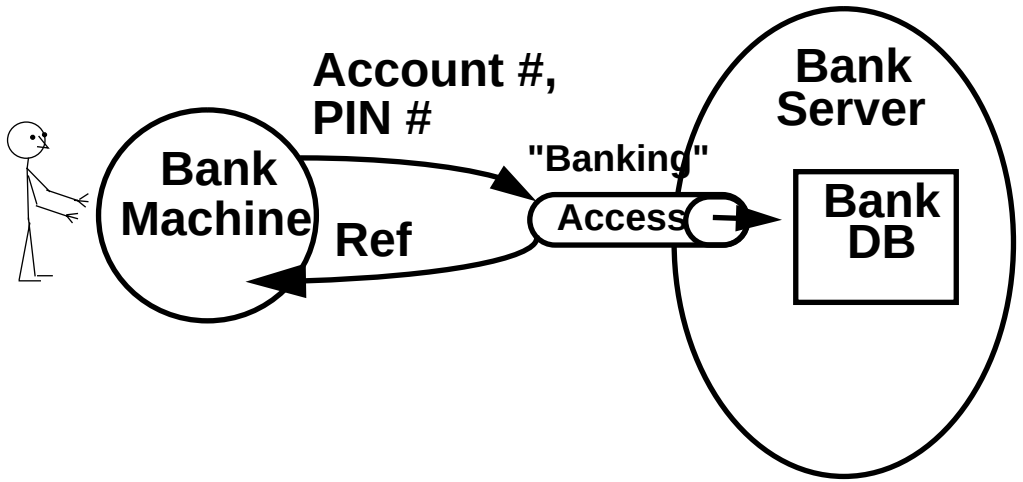
- *A Bank Machine (Automatic Teller Machine) system*
  - Users accessing their accounts
  - Bank managers administering these accounts

Bank Server - Interfaces

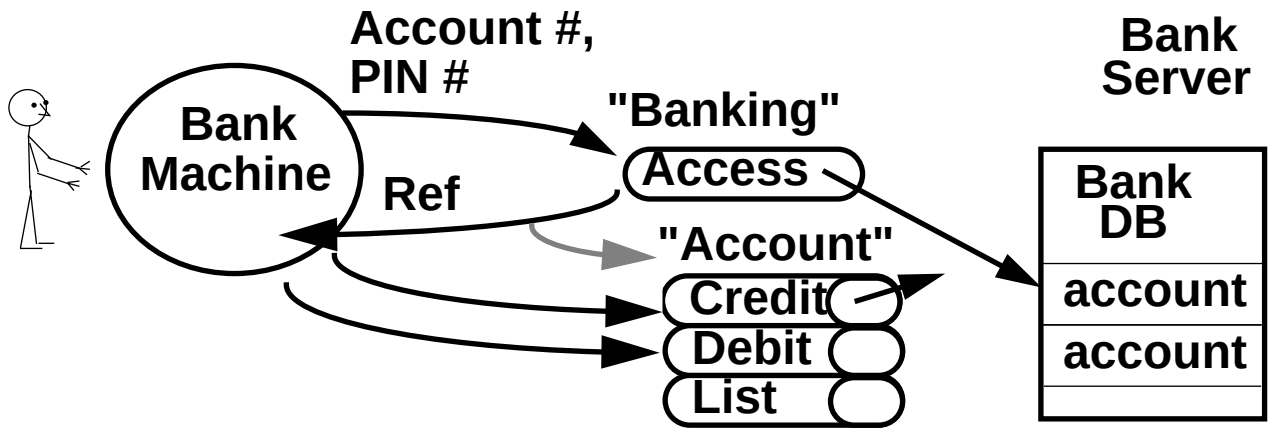


Banking Interface

- *User should only be able to interact with account if PIN is valid*
- *Do not want Accounts to be public*
  - customers should only access their own accounts
- *If "Access" succeeds, "Account" is created and its object reference returned*



Banking and Account interfaces



- *If "Access" succeeds, "Account" is created, and its object reference returned*
- *"Account" interface provides operations on one particular account*
- *Only that client knows the "Account" object reference*

Simple Bank Interfaces

- ***Banking (SBank)***
  - **for access with PIN by customer via Bank Machine, with PIN**
- ***Account (Account)***
  - **for credit/debit/list by customer via Bank Machine**
- ***Bank Management (SBankMgmt)***
  - **for create/destroy/list by manager**

Simple Bank - general definitions

```
typedef unsigned long AccountNumber;
typedef unsigned long PersonalIdentificationNumber;

exception NoSuchAccount {};
exception InvalidPin {};
exception InsufficientFunds {};
exception ResourcesExhausted {};
exception StaleAccountReference {};
```

Account Interface

- *Account - credit/debit/list account*
- *An Account object is created (via the SBank interface's Access operation) for each account being accessed*
- *So there is no need to pass account numbers as arguments to the operations in the Account interface*

Simple Bank - interface Account

```
interface Account
{
   struct AccountRecord {
      string owner;
      float balance;
      string lastaccess;
   };
   void Credit(in float Amount) raises(StaleAccountReference);
   void Debit(in float Amount)
       raises (InsufficientFunds, StaleAccountReference);
   void List(out AccountRecord List_R1)
       raises(StaleAccountReference);
};
```

Banking (SBank) Interface

- ***SBank - PIN based access to Account interfaces***
- ***Returns an Account interface for the account identified by an account number and corresponding PIN***
- ***Equivalent to the user interface offered by a real bank machine to a human user***

Simple Bank - interface SBank

```
interface SBank
{
      Account Access(in AccountNumber acct,
                     in PersonalIdentificationNumber pin)
         raises(NoSuchAccount, InvalidPin, ResourcesExhausted);
};
```

Bank Management (SBankMgmt) Interface

- ***SBankMgmt - management operations for:***
  - **creating an account**
  - **destroying an account**
  - **listing an account**
  - **listing all accounts**
- ***Note the use of an IDL sequence to return a variable amount of data when listing all accounts***

Simple Bank - interface SBankMgmt: types

```
interface SBankMgmt
{
    struct CreateRecord {
        AccountNumber acct;
        PersonalIdentificationNumber pin;
    };
    struct FullRecord {
        AccountNumber acct;
        PersonalIdentificationNumber pin;
        string owner;
        float balance;
        string lastaccess;
    };
    typedef sequence<FullRecord> ListAllResult;
```

Simple Bank - interface SBankMgmt: operations

```
   void Create(in string owner,
               in float balance, out CreateRecord Create_R1)
      raises(ResourcesExhausted);
   void Destroy(in AccountNumber acct) raises(NoSuchAccount);
   void ListOne(in AccountNumber acct, out FullRecord ListOne_R1)
      raises(NoSuchAccount);
   void ListAll(out ListAllResult ListAll_R1);
};
```

Summary

- ***Architectural ideas together with the appropriate software can be used to build distributed applications***
- ***ANSAware Reference Manual from ANSA/APM***