Advanced Software Engineering Report

Internet Banking System

Computer and Information Science

Master’s Course

02T0002   Daisuke Terasaki
1. Informal Specification

In the Internet banking system, the following resources are requested for self-account.

- The deposit account
- The savings account
- The deposit account of foreign currency
- The savings account of foreign currency

The deposit account can deposit and withdraw money without limitation. The saving account is can deposit money, and cannot withdraw money from the internet. And the remainder is an account for foreign currency. When money is transferred to foreign currency account, the following foreign currency rate is applied.

- The foreign currency rate

And the following resource is requested for investment trust.

- The investment trust account

To transfer money to the investment trust is equal to investing money through the investment fund.

And the following resource is requested for the list of public utility charges account registered into the customer.

- The public utility charges account list

If the account is registered into the public utility charges account list, a public utility charge will be charged directly automatically.

The following resources are required of customer information.

- name
- account number
- password

An account number is a number that determines the customer in only one clearly defined path.

The following resources are required for an account.

- The balance
- The payment details record

The payment details record is the history of the past transfer to self-account from others-account and to others-account from self-account.

In the internet banking system, the following functions must be provided.

- Refer a balance
- Refer payment details record

These functions get a balance or payment details record from each account.
Transfer a deposit account
Money is transferred from self deposit account to others deposit account.

Transfer a savings account
Money can be transferred to savings account from deposit account. However, money cannot be transferred from savings account.

Transfer a deposit account of foreign currency
Money is transferred from deposit account to deposit account of foreign currency with foreign currency rate. And then, money can be transferred to deposit account of foreign currency.

Transfer a savings account of foreign currency
Money is transferred from deposit account to savings account of foreign currency with foreign currency rate. And then, money cannot be transferred from savings account of foreign currency.

In the transfer functions, since the balance of the account does not become negative, if it is more transferred money than the balances of the account, an error returns.

In order to have the account of foreign currency, it is necessary to create an account by the following functions.

Open a deposit account of foreign currency
Open a savings account of foreign currency

The following resources are required for an investment trust control.

Open an investment trust account
Buy an investment trust account
Refer an investment trust account
Terminate an investment trust account

The open function is creating an investment trust account. The buy function is transferring money to an investment trust account, and is performing an investment through an investment fund. The refer function is getting an investment record. And the terminate function is removing an investment trust account.

And then, a public utility charges account can be registered by the following function.

Register a public utility charges

By registering to a public utility charges account, a charge is automatically paid from a deposit account.
2. Semi-formal Specification

module System_InternetBanking:

type
    Customer = composed of
        name
        account_number
        password
        deposit_account
        savings_account
        foreign_deposit_account
        foreign_savings_account
        investment_trust_account
        public_utility_charges_list
        e-mail;

    Account  = composed of
        balance
        payment_details

cdfd: none:

var
    customer_list: seq of Customer;
    foreign_currency_rate: real;

process Authenticate(customer_account_number, password)
    ext customer_list
    pre the account number exist on the customer list.
    post this process output control flow for authentication if input password equal to the
    password on the customer account.
end_process;

process Refer_Deposit_Payment_Details(customer_account_number) payment_details
    ext customer_list
    pre the account number exist on the customer list and the deposit account exist in it.
post the payment details of the deposit account is returned.
end_process;

process Refer_Savings_Payment_Details(customer_account_number) payment_details
  ext  customer_list
  pre  the account number exist on the customer list and the savings account exist in it.
  post the payment details of the savings account is returned.
end_process;

process Refer_Foreign_Deposit_Payment_Details(customer_account_number) payment_details
  ext  customer_list
  pre  the account number exist on the customer list and the deposit account of foreign currency
       exist in it.
  post the payment details of the deposit account of foreign currency is returned.
end_process;

process Refer_Foreign_Savings_Payment_Details(customer_account_number) payment_details
  ext  customer_list
  pre  the account number exist on the customer list and the savings account of foreign currency
       exist in it.
  post the payment details of the savings account of foreign currency is returned.
end_process;

process Refer_Deposit_Balance(customer_account_number) balance
  ext  customer_list
  pre  the account number exist on the customer list and the deposit account exist in it.
  post the deposit balance of the customer is returned.
end_process;

process Refer_Savings_Balance(customer_account_number) balance
  ext  customer_list
  pre  the account number exist on the customer list and the savings account exist in it.
  post the savings balance of the customer is returned.
end_process;

process Refer_Foreign_Currency_Rate() rate
  ext  foreign_currency_rate
  post the foreign currency rate is returned.
end_process;
process Refer_Foreign_Deposit_Balance(customer_account_number) balance
   ext  customer_list
   pre  the account number exist on the customer list and the deposit account of foreign currency exist in it.
   post the deposit balance of foreign currency of the customer is returned.
end_process;

process Refer_Foreign_Savings_Balance(customer_account_number) balance
   ext  customer_list
   pre  the account number exist on the customer list and the savings account of foreign currency exist in it.
   post the savings balance of foreign currency of the customer is returned.
end_process;

process Transfer_Deposit_Account(customer_account_number, money, transfer_account_number)
   ext  customer_list
   pre  the customer account number and transfer it exist on the customer list, and the deposit account exist in its and money is more many than a zero.
   post money is transferred from the deposit account of the customer to transfer deposit account and the result is written in the payment details if balance of the customer is more many than money, else money is not transferred to transfer deposit account and error maggage transfer to the customer.
end_process;

process Transfer_Savings_Account(customer_account_number, money)
   ext  customer_list
   pre  the customer account number exist on the customer list, and the deposit account and the savings account exist in the customer account and money is more many than a zero.
   post money is transferred from the deposit account to the savings account and the result is written in the payment details if balance of the deposit account of the customer is more many than money, else money is not transferred to the savings account and error maggage transfer to the customer.
end_process;

process Transfer_Foreign_Deposit_Account(customer_account_number, money)
   ext  customer_list
   pre  the customer account number exist on the customer list, and the deposit account and the deposit account of foreign currency exist in the customer account and money is more many
than a zero.

post money is transferred from the deposit account to the deposit account of foreign currency by the specific foreign currency rate and the result is written in the payment details if balance of the deposit account of the customer is more many than money. else money is not transferred to the deposit account of foreign currency and error maggage transfer to the customer.
end_process;

process Transfer_Foreign_Savings_Account(customer_account_number, money)
  ext customer_list
  pre the customer account number exist on the customer list, and the deposit account and the savings account of foreign currency exist in the customer account and money is more many than a zero.
  post money is transferred from the deposit account to the savings account of foreign currency by the specific foreign currency rate and the result is written in the payment details if balance of the deposit account of the customer is more many than money. else money is not transferred to the savings account of foreign currency and error maggage transfer to the customer.
end_process;

process Open_Foreign_Deposit_Account(customer_account_number)
  ext customer_list
  pre the customer account number exist on the customer list.
  post the deposit account of foreign currency is created if the deposit account of foreign currency does not exist on the customer account, else error maggage transfer to the customer.
end_process;

process Open_Foreign_Savings_Account(customer_account_number)
  ext customer_list
  pre the customer account number exist on the customer list.
  post the savings account of foreign currency is created if the savings account of foreign currency does not exist on the customer account, else error maggage transfer to the customer.
end_process;

process Register_Public_Utility_Charges_Account(customer_account_number,
                                               registration_account_number)
  ext customer_list
  pre the customer account number exist on the customer list.
  post registration account is registered on the customer account if registration account exist on the customer list and is not registered on the customer account, else error maggage transfer to customer.
end_process:

process Open_Investment_Trust_Account(customer_account_number)
  ext customer_list
  pre the customer account number exist on the customer list.
  post the investment trust account is created if the investment trust account does not exist on the customer account, else error maggage transfer to the customer.
end_process:

process Buy_Investment_Trust_Account(customer_account_number, money)
  ext customer_list
  pre the customer account number exist on the customer list, and the deposit account and the investment trust account exist in the customer account and money is more many than a zero.
  post money is transferred from the deposit account to the investment trust account and the result is written in the payment details if balance of the deposit account of the customer is more many than money, else money is not transferred to the investment account and error maggage transfer to the customer.
end_process:

process Refer_Investment_Trust_Details(customer_account_number, payment_details)
  ext customer_list
  pre the customer account number exist on the customer list and the investment trust account exist in it.
  post the payment details of the investment trust account is returned.
end_process:

process Terminate_Investment_Trust_Account(customer_account_number)
  ext customer_list
  pre the customer account number exist on the customer list, and the deposit account and the investment trust account exist in the customer account.
  post money is transferred from the investment trust account to the deposit account and then investment trust account is closed.
end_process:

end_module:
3. Formal Specification

module System_InternetBanking:

type
  Account  = composed of
    balance: nat
    payment_details: seq of Details
  end;

  Details  = composed of
    abstract: string
    destination: string
    payment: nat
    deposit: nat
    balance: nat
  end;

  Utility  = set of Customer;

  Customer = composed of
    name: string  /* no use */
    account_number: nat
    password: string
  end;

  Accounts = map Customer to Account;
  Utilities = map Customer to Utility;

cdfd: Figure 1:

var
  deposit_account: Accounts;
  savings_account: Accounts;
  foreign_deposit_account: Accounts;
process Get_Customer(account_number: nat) customer: Customer | error_message: string
  ext rd deposit_account
  post exists[c inset dom(deposit_account) |
    c.account_number = account_number & customer = c]
  or not exists[c inset dom(deposit_account) |
    c.account_number = account_number & error_message = “this account is not found”]
  comment
    a deposit account is considered a database, and if a account number exists in database,
    then the customer information that contains the account number is returned, else an error
    message is returned.
end_process;

process Authenticate(c: Customer, password: string)
  customer: Customer | error_message: string
  ext rd deposit_account
  pre c inset dom(deposit_account)
  post if c.password = password
    then customer = c
    else error_message = “authentication error”
  comment
    a deposit account is considered a database, and if a customer’s password corresponds to
    the inputting password, the customer information is returned, else an error message is
    returned.
end_process;

process Refer_Foreign_Currency_Rate() rate: real
  ext rd foreign_currency_rate
  post foreign_currency_rate = rate
end_process;

process Self_Account_Control(customer: Customer | customer: Customer, money: nat)
  balance: nat0 | details: seq of Details |
  signal: void | error_message: string
ext wr deposit_account
    wr savings_account
    wr foreign_deposit_account
    wr foreign_savings_account
    rd foreign_currency_rate
deocom System_SelfAccountControl
comment
    The composition of process to solve only self-account
end_process;

process Transfar_To_Others(customer: Customer, money: nat, transfer_account: Customer)
signal: void | error_message: string
ext wr deposit_account
pre customer inset dom(deposit_account)
deocom System_TransfarToOthers
comment
    The composition of process to transfer money to the others-account
end_process;

process Public_Utility_Charges_Control(customer: Customer, registration_account: Customer)
signal: void | error_message: string
ext rd deposit_account
    wr public_utility_charges_list
pre customer inset dom(deposit_account)
deocom System_PublicUtilityChargesControl
comment
    The composition of process to control the public utility charges
end_process;

process Investment_Trust_Control(customer: Customer | customer: Customer, money: nat)
    balance: nat0 | details: seq of Details | signal: void |
    error_message: string
ext wr deposit_account
    wr investment_trust_account
pre customer inset dom(deposit_account)
deocom System_InvestmentTrustControl
comment
    The composition of process to control the investment trusts
end_process;
function create_payment_details(old_details: seq of Details, abstract: string, customer: Customer, money: nat, balance: nat)
    details: seq of Details
    post details = cons(old_details,
                        [mk_Details(abstract, customer.name, money, 0, balance)])
    /* a helper function for the payment details record when money was paid to others-account from self-account */
end_function;

function create_deposit_details(old_details: seq of Details, abstract: string, customer: Customer, money: nat, balance: nat0)
    details: seq of Details
    post details = cons(old_details, [mk_Details(abstract, customer.name, 0, money, balance)])
    /* a helper function for the payment details record when money was paid to self-account from others-account */
end_function;

end_module;

module System_SelfAccountControl / System_InternetBanking:

    cdfd: Figure 2:

    var
        deposit_account: Accounts;
        savings_account: Accounts;
        foreign_deposit_account: Accounts;
        foreign_savings_account: Accounts;
        foreign_currency_rate: real;

    process Refer_Balance(customer: Customer) balance: nat0 | error_message: string
        ext rd deposit_account
        rd savings_account
        rd foreign_deposit_account
        rd foreign_savings_account
The composition of process to get the balance on the self-account

process Transfar_To_Self(customer: Customer, money: nat)
    signal: void | error_message: string

    ext wr deposit_account
    wr savings_account
    wr foreign_deposit_account
    wr foreign_savings_account
    rd foreign_currency_rate

    pre customer inset dom(deposit_account)
    decom System_TransfarToSelf
    comment
    The composition of process to transfer money to the self-account
end_process;

process Refer_Payment_Details(customer: Customer)
    details: seq of Details | error_message: string

    ext rd deposit_account
    rd savings_account
    rd foreign_deposit_account
    rd foreign_savings_account

    decom System_ReferPaymentDetails
    comment
    The composition of process to get the payment details record on the self-account
end_process;

process Open_Foreign_Account(customer: Customer)
    signal: void | error_message: string

    ext rd deposit_account
    wr foreign_deposit_account
    wr foreign_savings_account

    pre customer inset dom(deposit_account)
    decom System_OpenForeignAccount
    comment
    The composition of process to open the foreign currency account on the self-account
module System_TransferToOthers / System_InternetBanking:

cdfd: Figure 3:

var
deposit_account: Accounts;

process Transfer_Deposit_Account(customer: Customer, money: nat, transfer_account: Customer)
signal: void | error_message: string
ext wr deposit_account
pre customer inset dom(deposit_account) and
post if transfer_account inset dom(~deposit_account)
then if ~deposit_account(customer).balance >= money
then deposit_account =
override(~deposit_account,
{customer->modify(~deposit_account(customer),
balance->~deposit_account(customer).balance - money,
payment_details->create_payment_details(
"deposit_account(customer).payment_details,
"transfar deposit account",
transfer_account,
money,
"deposit_account(customer).balance - money)))
{transfer_account->modify(~deposit_account(transfer_account),
balance->~deposit_account(transfer_account).balance + money,
payment_details->create_deposit_details(
"deposit_account(transfer_account).payment_details,
"transfar deposit account",
customer,
money,
"deposit_account(transfer_account).balance + money)))
and bound(signal)
else error_message = "balance of this account is more many than money"
else error_message = "transfer account is not found"

comment

If the customer and the transfer account exist on the deposit account and it is more
inputting money than the customer’s deposit balances. then money is transferred from the
customer’s deposit account to the transfer account, else error message is returned.
end_process:

end_module:

module System_PublicUtilityChargesControl / System_InternetBanking:

cdfd: Figure 4:

var

  deposit_account: Accounts;

process Register_Public_Utility_Charges_Account(customer: Customer,
registration_account: Customer)
  signal: void | error_message: string

  ext rd deposit_account
  wr public_utility_charges_list
  pre customer inset dom(deposit_account)
  post if registration_account inset dom(deposit_account)
    then if not exists[p inset rng(~public_utility_charges_list) |
      p = registration_account]
      then public_utility_charges_list =
        override(~public_utility_charges_list,
        {customer->union(rng(~public_utility_charges_list),
        {registration_account})
        and bound(signal)
      else error_message = "this public utility charges account exist already"
      else error_message = "registration account is not found"

      comment

      If the customer and the registration account exist on the deposit account and the
      registration account is not registered in the customer’s account, then the registration
      account is registered in the customer’s account, else error message is returned:
module System_InvestmentTrustControl / System_InternetBanking:

cdfd: Figure 5:

var
    deposit_account: Accounts;
    investment_trust_account: Accounts;

process Open_Investment_Trust_Account(customer: Customer)
    signal: void | error_message: string
    ext rd deposit_account
    wr investment_trust_account
    pre customer inset dom(deposit_account)
    post if not customer inset dom(~investment_trust_account)
        then investment_trust_account =
            marge(~investment_trust_account
                {customer->mk_Account(0, [mk_Details("open investment trust account", 
                                ",", 0, 0, 0)])})
            and bound(signal)
        else error_message = "investment trust account exist already"
    comment
        If the customer exist on the deposit account and the customer does not exist on the
        investment trust account, then customer’s investment trust account is created. else error
        message is returned;
end_process:

process Buy_Investment_Trust_Account(customer: Customer, money: nat)
    signal: void | error_message: string
    ext wr deposit_account
    wr investment_trust_account
    pre customer inset dom(deposit_account)
    post if customer inset dom(~investment_trust_account)
        then if ~deposit_account(customer).balance >= money
then deposit_account =
    override(`deposit_account,
        {customer->modify(`deposit_account(customer),
            balance->`deposit_account(customer).balance - money,
            payment_details->create_payment_details(`
                `deposit_account(customer).payment_details,
                "buy investment truct",
                customer,
                money,
                `deposit_account(customer).balance - money))}
    )
and investment_trust_account =
    override(`investment_trust_account,
        {customer->modify(`investment_trust_account(customer),
            balance->`investment_trust_account(customer).balance + money,
            payment_details->create_deposit_details(`
                `investment_trust_account(customer).payment_details,
                "buy investment truct",
                customer,
                money,
                `investment_trust_account(customer).balance + money))}
    )
and bound(signal)
else error_message = "balance of this account is more many than money"
else error_message = "investment trust account is not found"

comment
    If the customer exist on the deposit account and the investment trust account, and it is
    more inputting money than the customer’s deposit balances, then money is transferred from
    the deposit account to the investment trust account. else error message is returned.
end_process;

process Refer_Investment_Trust_Details(customer: Customer)
    details: seq of Details | error_message: string
ext rd investment_trust_account
post if customer inset dom(investment_trust_account)
    then investment_trust_account(customer).payment_details = details
    else error_message = "investment trust account is not found"
end_process;

process Terminate_Investment_Trust_Account(customer: Customer)
signal: void | error_message: string

ext wr deposit_account
    wr investment_trust_account
pre customer inset dom(deposit_account)
post if customer inset dom(~investment_trust_account)
    then deposit_account =
        override(~deposit_account,
            {customer->modify(~deposit_account(customer),
                balance->~deposit_account(customer).balance +
                investment_trust_account(customer).balance,
                payment_details->create_deposit_details(
                    ~deposit_account(customer).payment_details,
                    “terminate investment trust”,
                    customer,
                    money,
                    ~deposit_account(customer).balance +
                    investment_trust_account(customer).balance))}
        and investment_trust_account = drstb({customer}, ~investment_trust_account)
        and bound(signal)
    else error_message = “investment trust account is not found”

comment
    If the customer exists on the deposit account and the investment trust account, then money is transferred from the deposit account to the investment trust account and then customer’s investment trust account is removed, else error message is returned.
end_process;

end_module:

module System_ReferBalance / System_SelfAccountControl:

cddf: Figure 6:

var
    deposit_account: Accounts;
    savings_account: Accounts;
    foreign_deposit_account: Accounts;
    foreign_savings_account: Accounts;
process Refer_Deposit_Balance(customer: Customer) balance: nat0
  ext rd deposit_account
  pre customer inset dom(deposit_account)
  post deposit_account(customer).balance = balance
end_process;

process Refer_Savings_Balance(customer: Customer)
  balance: nat0 | error_message: string
  ext rd savings_account
  post if customer inset dom(savings_account)
        then savings_account(customer).balance = balance
           else error_message = "savings account is not found"
end_process;

process Refer_Foreign_Deposit_Balance(customer: Customer)
  balance: nat0 | error_message: string
  ext rd foreign_deposit_account
  post if customer inset dom(foreign_deposit_account)
        then foreign_deposit_account(customer).balance = balance
           else error_message = "deposit account of foreign currency is not found"
end_process;

process Refer_Foreign_Savings_Balance(customer: Customer)
  balance: nat0 | error_message: string
  ext rd foreign_savings_account
  post if customer inset dom(foreign_savings_account)
        then foreign_savings_account(customer).balance = balance
           else error_message = "savings account of foreign currency is not found"
end_process;

end_module;

module System_TransfarToSelf / System_SelfAccountControl;

cdfd: Figure 7:
var
deposit_account: Accounts;
savings_account: Accounts;
foreign_deposit_account: Accounts;
foreign_savings_account: Accounts;
foreign_currency_rate: real;

process Transfer_Savings_Account(customer: Customer, money: nat)
    signal: void | error_message: string
ext wr deposit_account
    wr savings_account
pre customer inset dom(deposit_account)
post if customer inset dom(~savings_account)
    then if ~deposit_account(customer).balance >= money
        then deposit_account =
            override(~deposit_account,
                {customer->modify(~deposit_account(customer),
                    balance->~deposit_account(customer).balance - money,
                    payment_details->create_payment_details(~
                        deposit_account(customer).payment_details,
                        "transfar savings account",
                        customer,
                        money,
                        ~deposit_account(customer).balance - money)})
            and savings_account =
            override(~savings_account,
                {customer->modify(~savings_account(customer),
                    balance->~savings_account(customer).balance + money,
                    payment_details->create_deposit_details(~
                        savings_account(customer).payment_details,
                        "transfar savings account",
                        customer,
                        money,
                        ~savings_account(customer).balance + money)})
            and bound(signal)
        else error_message = "balance of this account is more many than money"
    else error_message = "savings account is not found"
comment
If the customer exist on the deposit account and the savings account, and it is more
inputting money than the customer’s deposit balances, then money is transferred from the deposit account to the savings account, else error message is returned.

end_process:

process Transfer_Foreign_Deposit_Account(customer: Customer, money: nat)
    signal: void | error_message: string

    ext wr deposit_account
    wr foreign_deposit_account
    rd foreign_currency_rate

    pre customer inset dom(deposit_account)
    post if customer inset dom(~foreign_deposit_account)
        then if ~deposit_account(customer).balance >= money
            then deposit_account =
                override(~deposit_account,
                    {customer->modify(~deposit_account(customer),
                        balance->(~deposit_account(customer).balance - money) * foreign_currency_rate,
                        payment_details->create_payment_details(~deposit_account(customer).payment_details,
                            "transfar deposit account of foreign currency", customer,
                            money,
                            (~deposit_account(customer).balance - money) * foreign_currency_rate))})

            and foreign_deposit_account =
                override(~foreign_deposit_account,
                    {customer->modify(~foreign_deposit_account(customer),
                        balance->(~foreign_deposit_account(customer).balance + money) * foreign_currency_rate,
                        payment_details->create_deposit_details(~foreign_deposit_account(customer).payment_details,
                            "transfar deposit account of foreign currency", customer,
                            money,
                            (~foreign_deposit_account(customer).balance + money) * foreign_currency_rate))})

            and bound(signal)
        else error_message = "balance of this account is more many than money"
    else error_message = "deposit account of foreign currency is not found"
If the customer exist on the deposit account and the deposit account of foreign currency, and it is more inputting money than the customer’s deposit balances, then money is transferred from the deposit account to the deposit account of foreign currency with the foreign currency rate, else error message is returned.

end_process;

process Transfer_To_Deposit_From_Foreign_Deposit(customer: Customer, money: nat)
    signal: void | error_message: string
    ext wr deposit_account
    wr foreign_deposit_account
    rd foreign_currency_rate
    pre customer inset dom(deposit_account)
    post if customer inset dom(~foreign_deposit_account)
        then if ~deposit_account(customer).balance >= money
            then deposit_account =
                override(~deposit_account,
                    {customer->modify(~deposit_account(customer),
                        balance->(~deposit_account(customer).balance + money) *
                        (1 / foreign_currency_rate),
                        payment_details->create_deposit_details(~deposit_account(customer).payment_details,
                            "transfer deposit account", customer,
                            money,
                            (~deposit_account(customer).balance + money) *
                            (1 / foreign_currency_rate)))}
            and foreign_deposit_account =
                override(~foreign_deposit_account,
                    {customer->modify(~foreign_deposit_account(customer),
                        balance->(~foreign_deposit_account(customer).balance - money) *
                        (1 / foreign_currency_rate),
                        payment_details->create_payment_details(~foreign_deposit_account(customer).payment_details,
                            "transfer deposit account", customer,
                            money,
                            (~foreign_deposit_account(customer).balance - money) *
                            (1 / foreign_currency_rate)))}
and bound(signal)
else error_message = “balance of this account is more many than money”
else error_message = “deposit account of foreign currency is not found”

comment
If the customer exist on the deposit account and the deposit account of foreign currency, and it is more inputting money than the customer’s deposit balances of foreign currency, then money is transferred from the deposit account of foreign currency to the deposit account with foreign currency rate. else error message is returned.
end_process:

process Transfer_Foreign_Savings_Account(customer: Customer, money: nat)
    signal: void | error_message: string
    ext wr deposit_account
    wr foreign_savings_account
    rd foreign_currency_rate
    pre customer inset dom(deposit_account)
    post if customer inset dom(~foreign_savings_account)
then if ~deposit_account(customer).balance >= money
then deposit_account =
    override(~deposit_account,
        {customer->modify(~deposit_account(costomer),
            balance->(~deposit_account(costomer).balance - money) * foreign_currency_rate,
            payment_details->create_payment_details(~deposit_account(costomer).payment_details,
                “transfar savings account of foreign currency”,
                customer,
                money,
                (~deposit_account(costomer).balance - money) * foreign_currency_rate))}
    and foreign_savings_account =
    override(~foreign_savings_account,
        {customer->modify(~foreign_savings_account(costomer),
            balance->(~~foreign_savings_account(costomer).balance + money) * foreign_currency_rate,
            payment_details->create_deposit_details(~foreign_savings_account(costomer).payment_details,
                “transfar savings account of foreign currency”,
                customer,
                money,
                (~foreign_savings_account(costomer).balance + money) * foreign_currency_rate))}
customer,
money,

("foreign_savings_account(customer).balance + money) *

and bound(signal)

else error_message = "balance of this account is more many than money"
else error_message = "savings account of foreign currency is not found"

comment

If the customer exist on the deposit account and the savings account of foreign currency,
and it is more inputting money than the customer’s deposit balances, then money is transferred
from the deposit account to the savings account of foreign currency with foreign currency rate,
else error message is returned.
end_process;

end_module;

module System_RefferPaymentDetails / System_SelfAccountControl;

cdfd: Figure 8;

var

deposit_account: Accounts;
savings_account: Accounts;
foreign_deposit_account: Accounts;
foreign_savings_account: Accounts;

process Refer_Deposit_Payment_Details(customer: Customer) details: seq of Details
ext rd deposit_account
pre customer inset dom(deposit_account)
post deposit_account(customer).payment_details = details
end_process;

process Refer_Savings_Payment_Details(customer: Customer)

details: seq of Details | error_message: string
ext rd savings_account
post if customer inset dom(savings_account)
then savings_account(customer).payment_details = details
else error_message = "savings account is not found"
end_process;

process Refer_Foreign_Deposit_Payment_Details(customer: Customer)

details: seq of Details |
error_message: string

ext  rd foreign_deposit_account

post if  customer inset dom(foreign_deposit_account)
    then foreign_deposit_account(customer).payment_details = details
    else error_message = "deposit account of foreign currency is not found"
end_process;

process Refer_Foreign_Savings_Payment_Details(customer: Customer)

details: seq of Details |
error_message: string

ext  rd foreign_savings_account

post if  customer inset dom(foreign_savings_account)
    then foreign_savings_account(customer).payment_details = details
    else error_message = "savings account of foreign currency is not found"
end_process;

end_module:

module System_OpenForeignAccount / System_SelfAccountControl:

cdfd: Figure 9:

var

   deposit_account: Accounts;
   foreign_deposit_account: Accounts;
   foreign_savings_account: Accounts;

process Open_Foreign_Deposit_Account(customer: Customer)

    signal: void | error_message: string

    ext  rd deposit_account

    wr foreign_deposit_account

    pre  customer inset dom(deposit_account)
post if not customer inset dom("foreign_deposit_account")
then foreign_deposit_account =
  marge("foreign_deposit_account
   {customer->mk_Account(0,
      [mk_Details("open deposit account of foreign currency",
         "", 0, 0, 0)])})
   and bound(signal)
else error_message = "deposit account of foreign currency exist already"
comment
  If the customer exists on the deposit account and the customer does not exist on the deposit
  account of foreign currency, then customer’s deposit account of foreign currency is created,
else error message is returned:
end_process;

process Open_Foreign_Savings_Account(customer: Customer)
  signal: void | error_message: string
ext rd deposit_account
wr foreign_savings_account
pre customer inset dom(deposit_account)
post if not customer inset dom("foreign_savings_account")
then foreign_savings_account =
  marge("foreign_savings_account
   {customer->mk_Account(0,
      [mk_Details("open savings account of foreign currency",
         "", 0, 0, 0)])})
   and bound(signal)
else error_message = "savings account of foreign currency exist already"
comment
  If the customer exists on the deposit account and the customer does not exist on the savings
  account of foreign currency, then customer’s savings account of foreign currency is created,
else error message is returned:
end_process;
end_module;
Figure 2
Figure 3

Figure 4
Figure 5
Figure 6
Figure 7
Refer_Deposit_Payment_Details

Refer_Savings_Payment_Details

Refer_Foreign_Deposit_Payment_Details

Refer_Foreign_Savings_Payment_Details

Figure 8
Figure 9

The diagram illustrates the process of opening foreign savings and deposit accounts.

- Customer initiates the process to open a foreign savings account.
- Customer also initiates the process to open a foreign deposit account.
- There are signals and error messages associated with the processes.