

PROJECT

CS 586; Spring 2016

Deadlines:

MDA-EFSM: **April 6, 2016** (15% of the total project score)
After **April 10** the MDA-EFSM will not be accepted.

Final Project: **May 2, 2016**

Late submissions: 50% off

After **May 5** the final project will not be accepted.

This is an **individual** project not a team project.

The **hardcopy** of the project must be submitted. Electronic submissions are not acceptable. Notice that the Blackboard project submissions are only considered as a proof of submission on time (before the deadline).

Goal:

The goal of this project is to design two different ACCOUNT components using a Model-Driven Architecture (MDA) and then implement these ACCOUNT components based on this design.

Description of the Project:

There are two ACCOUNT components: ACCOUNT-1 and ACCOUNT-2:

ACCOUNT-1 component supports the following operations:

open (string p, string y, float a)	// open an account where p is a pin, y is an user's identification #, and a is a balance
pin (string x)	// provides pin #
deposit (float d);	// deposit amount d
withdraw (float w);	// withdraw amount w
balance ();	// display the current balance
login(string y)	// login where y is a client's identification #
logout()	// logout from the account
lock(string x)	// locks an account where x is a pin
unlock(string x)	// unlocks an account where x is a pin

ACCOUNT-2 component supports the following operations:

OPEN (int p, int y, int a)	// open an account where p is a pin, y is an user's identification #, and a is a balance
PIN (int x)	// provides pin #
DEPOSIT (int d);	// deposit amount d
WITHDRAW (int w);	// withdraw amount w
BALANCE ();	// display the current balance
LOGIN(int y)	// login where y is a client's identification #
LOGOUT()	// logout from the account
suspend()	// suspends an account
activate()	// activates a suspended account
close()	// an account is closed

Both ACCOUNT components are state-based components and support three types of transactions: withdrawal, deposit, and balance inquiry. Before any transaction can be performed, operation *open(p, y, a)* (or *OPEN(p, y, a)*) must be issued, where *y* is a client's identification #, *p* is a pin used to get permission to perform transactions and *a* is an initial balance in the account. It is assumed that *open()/OPEN()* operation is issued only once for a given account. Before any transaction can be performed, operation *login(y)* must be issued (where *y* is a client's identification #) followed by *pin(x)* (or *PIN(x)*) operation. The *pin(x)* (or *PIN(x)*) operation must contain the valid pin # that must be the same as the pin # provided in *open(p, y, a)* (or *OPEN(p, y, a)*) operation. There is a limit on the number of attempts with an invalid pin. The account can be overdrawn (below minimum balance), but a penalty may apply. If the balance is below the minimum balance then the withdrawal transaction cannot be performed. The account may become locked by *lock* operation or suspended by *suspend* operation. If the account is locked, withdrawal, deposit, logout and balance transactions cannot be performed. A locked account becomes unlocked by *unlock* operation. A suspended account can be activated by *activate* operation. In addition, a suspended account can be closed by *close* operation. The detailed behavior of both ACCOUNT components is specified using EFSM. The EFSM of Figure 1 shows the detail behavior of ACCOUNT-1, and the EFSM of Figure 2 shows the detailed behavior of ACCOUNT-2. Notice that there are several differences between both ACCOUNTs.

Aspects that vary between two ACCOUNT components:

- a. Maximum number of times incorrect pin can be entered
- b. Minimum balance
- c. Display menu(s)
- d. Messages, e.g., error messages, etc.
- e. Penalties
- f. Operation names and signatures
- g. Data types
- h. etc.

The goal of this project is to design two ACCOUNT components using a Model-Driven Architecture (MDA) covered in the course. An executable meta-model, referred to as MDA-EFSM, of ACCOUNT components should capture the “generic behavior” of both ACCOUNT components and should be de-coupled from data and implementation details. Notice that in your design there should be **ONLY** one MDA-EFSM for both ACCOUNT components. In addition, in the Model-Driven Architecture coupling between components should be minimized and cohesion of components should be maximized (components with high cohesion and low coupling between components). The meta-model (MDA-EFSM) used in the Model-Driven architecture should be expressed as an EFSM (Extended Finite State Machine) model. Notice that the EFSMs shown in Figures 1 and Figure 2 are **not acceptable** as a meta-model (MDA-EFSM) for this model driven architecture.

After the MDA-EFSM is created, you need to design and implement two ACCOUNT components using the model-driven architecture. In your design you **MUST** use the following OO design patterns:

- state pattern
- strategy pattern
- abstract factory pattern

SUBMISSIONS & DEADLINES

I. MDA-EFSM submission: April 6, 2016

MDA-EFSM model for the *ACCOUNT* components:

- A list of events for the MDA-EFSM
- A list of actions for the MDA-EFSM.
 - The responsibility of each action must be described.
- A state diagram of the MDA-EFSM
- Pseudo-code of all operations of Input Processors of *ACCOUNT-1* and *ACCOUNT-2*.

After **April 10, 2016** the MDA-EFSM will not be accepted.

II. Final Project submission: May 2, 2016

After **May 5** the final project will not be accepted.

The detailed description of the final project report and deliverables will be posted later on.