Solution 3

Quebec

Daniel Kostic, Guilherme Fial, Igor Fedotov, Stefan Vujovic

Exercise 1: Architectural Styles

a) Is it possible to consider Model-View-Controller (MVC) as layered architecture? Please explain. (1 point)

NO. MVC is a triangular and mostly unidirectional architecture (model->view->controller->back to model) and layered architecture is sequential bidirectional (data layer <-> business layer <-> presentation layer).

One of the base principles of *layered architecture* is that a layer uses only services from the immediate lower level (i.e., it acts as a client). It is not allowed to directly use layers below/above the immediate lower/higher one.

In contrast to layered architecture, **classic web-based** *MVC* pattern implies that the communication between “layers” is expected to be looped: the controller talks to the model, **the model has a reference back to the view**, which, in turn, talks to the controller. So, for example, it is possible for the model, after it changed its state, to push the changes to multiple views.

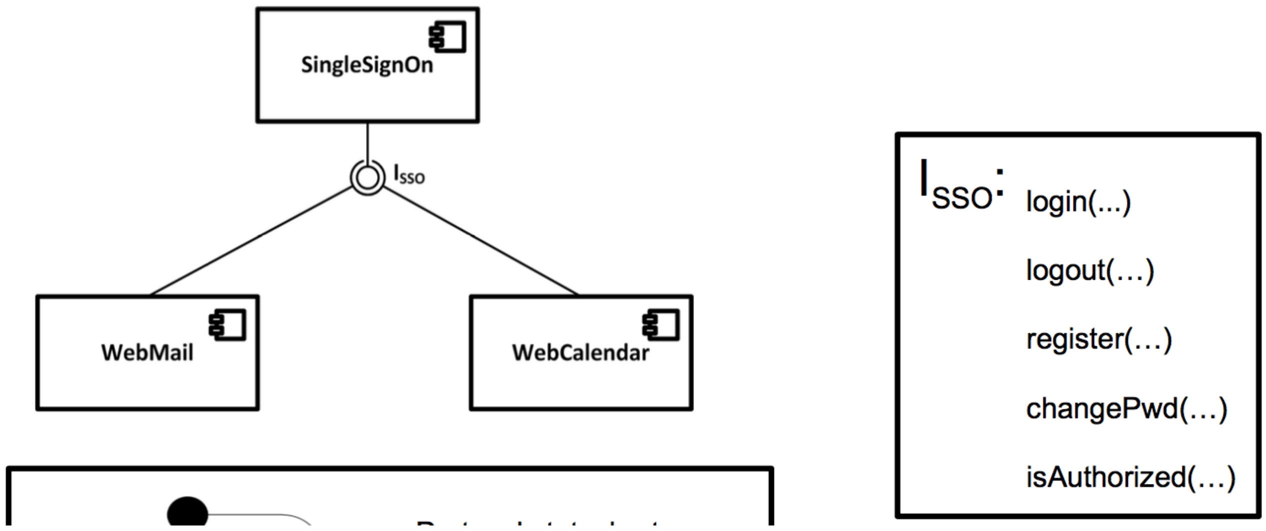
b) Please associate the layers of UWE with MVC. (1 point)

UWE	**MVC**
Content layer	Model
Hypertext layer	Controller
Presentation layer	View

c) What do you understand under the terms “high cohesion” and “low coupling”? Please explain. (1 point)

High cohesion means that the strength of relationship between elements within a given component is high. Such relationship can, for example, be reflected in logical of functional similarity of the methods within a component. Using the example from the lecture (pic 1c.1), we can see that the component *SingleSignOn* has all these methods to control the user auth data and nothing more, therefore we can say it has a high (logical and probably functional) cohesion.
Strong cohesion increases the maintainability of the system, because changes in one component will affect fewer other components (if not none) and will also require less changes in other components overall. It helps to simplify the component complexity throughout the whole project,

since the elements of the application will be split by their logic/functionality into distinct components.



Pic 1c.1 – SSO component within some system and its elements

Low coupling between 2 components implies that *component A* should have:

1. as few as possible dependencies on
2. as little as possible direct knowledge about

component B, to maintain the reusability ability to be modified without affecting the work of *component B*.
In general, if the architecture is loosely coupled – it has much less risk that some change within a component will cause instability or unexpected behaviour of other components. It allows to find out where exactly the problem occurred more promptly, thus dramatically simplifying testing. Also, low coupled system can be easily visualized by breaking itself down into distinct elements.

d) What kind of design pattern is used in Model-View-Controller? Please explain this pattern and describe where and how this pattern is used in MVC. (2 points)

The **"observer"** pattern is used in MVC. The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents (one-to-many relationship), called observers, and notifies them automatically of any state changes, usually by calling one of their methods. (source: wikipedia).
In MVC the model keeps reference to the views (the views observe the model) and notifies them about its state change. For example, there are 3 views that display pie chart, scatter plot and bar graph; when the model state changes, it notifies the views and they update the plots they are displaying without need to ask the model for updates.
Also, the view has a reference to the controller (the controller observes the view), to notify it about the user interactions with the view, so that the controller does appropriate function calls to the model.

Exercise 2: Modelling

Consider the online-shop “StorePub” which was defined in Exercise 2:

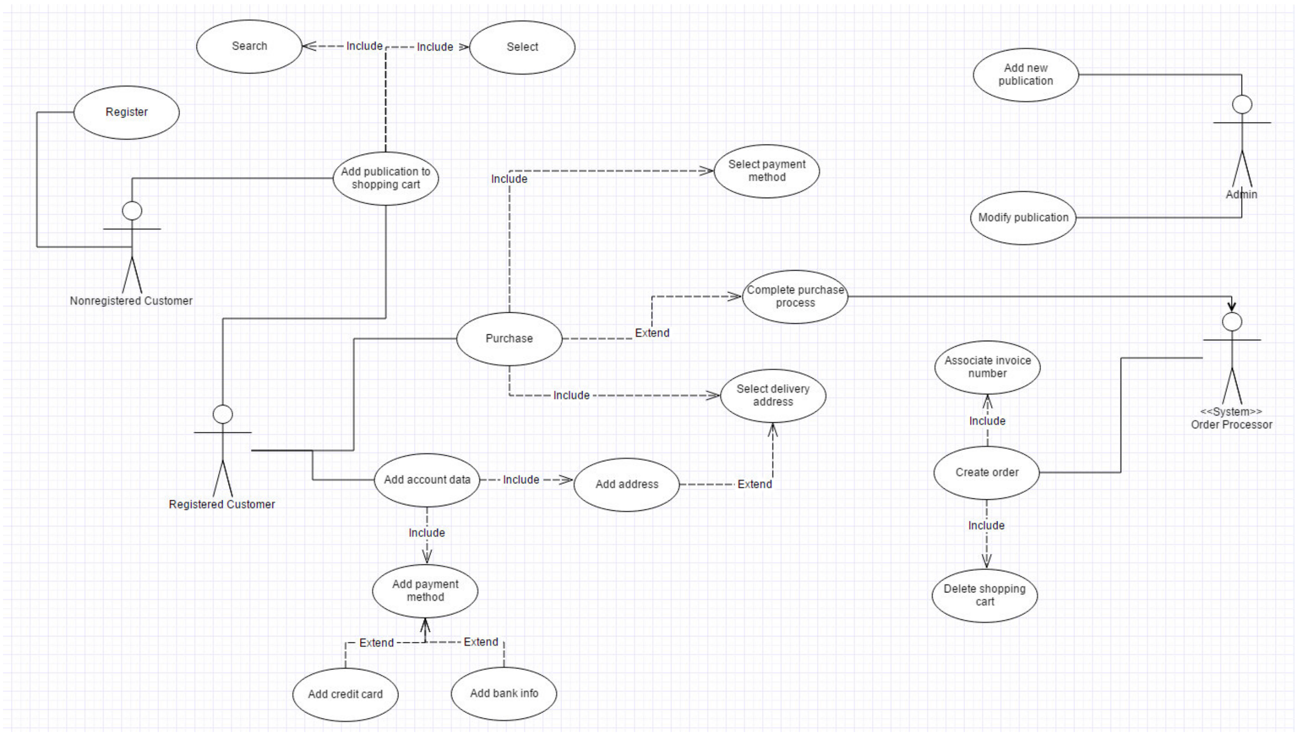
Application “StorePub” is an online shop for academic publications. StorePub has two kinds of user, namely Customer and Administrator. A customer could be Registered and NonRegistered. Moreover, StorePub has a Shopping Cart and a Catalog. Different publications are contained in a Catalog.

A NonRegistered customer can search and select publications, add publications to the shopping cart, and get registered. A Registered customer can also search and select publications, and add the publications to the shopping cart. A Registered customer must have at least one predefined Address. Moreover, a payment method for the Registered customer must be defined. A Registered customer can pay with a credit card or through a bank transfer. A Customer can add more than one publication to a shopping cart. Each publication in the shopping cart can be purchased. A publication may be a dissertation, a technical report or a lecture material.

The Administrator can add new publications to the catalog and modify existing publications’ information.

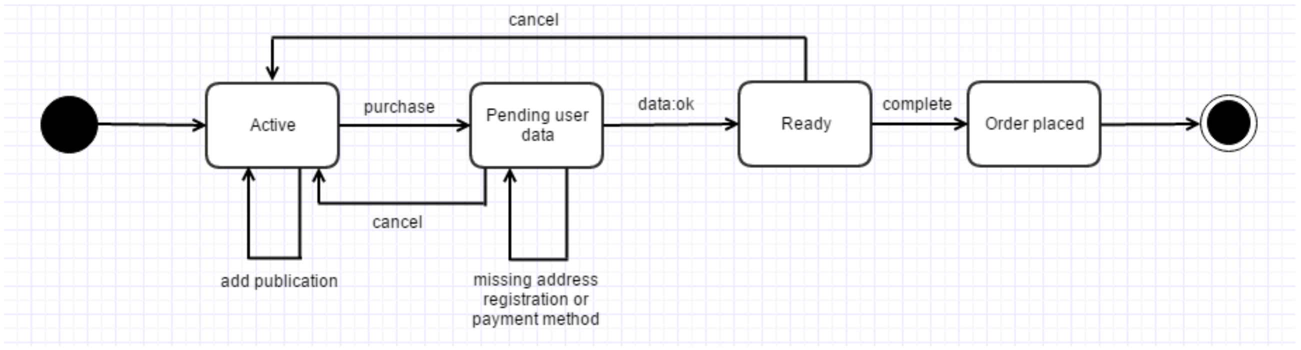
During the purchase process for an existing shopping cart, a Registered customer must provide a delivery address by selecting one of his predefined addresses or by adding a new address. Furthermore, the Registered customer must select a payment method. If the Customer completes the purchase process, StorePub creates an order for the actual shopping cart and deletes the shopping cart. An invoice number is associated with the order.

a) Describe the functional requirements of StorePub online-shop using a Use Case diagram. (2)



b) Describe the lifecycle of a shopping cart using a stateMachine diagram (UML). (3)

In the description it says that each publication can be purchased separately, but the purchase process is only described for the whole cart. It is confusing. But if a single publication purchase is possible, then we can add a *conditional* transition from an “**order placed**” state to “**active**” state. The condition would be: “if the shopping cart still contains publications after the order was placed”. But again, we are just guessing here, since this info is not provided.


cancel
Active
purchase
Pending user data
data:ok
Ready
complete
Order placed
cancel
add publication
missing address registration or payment method