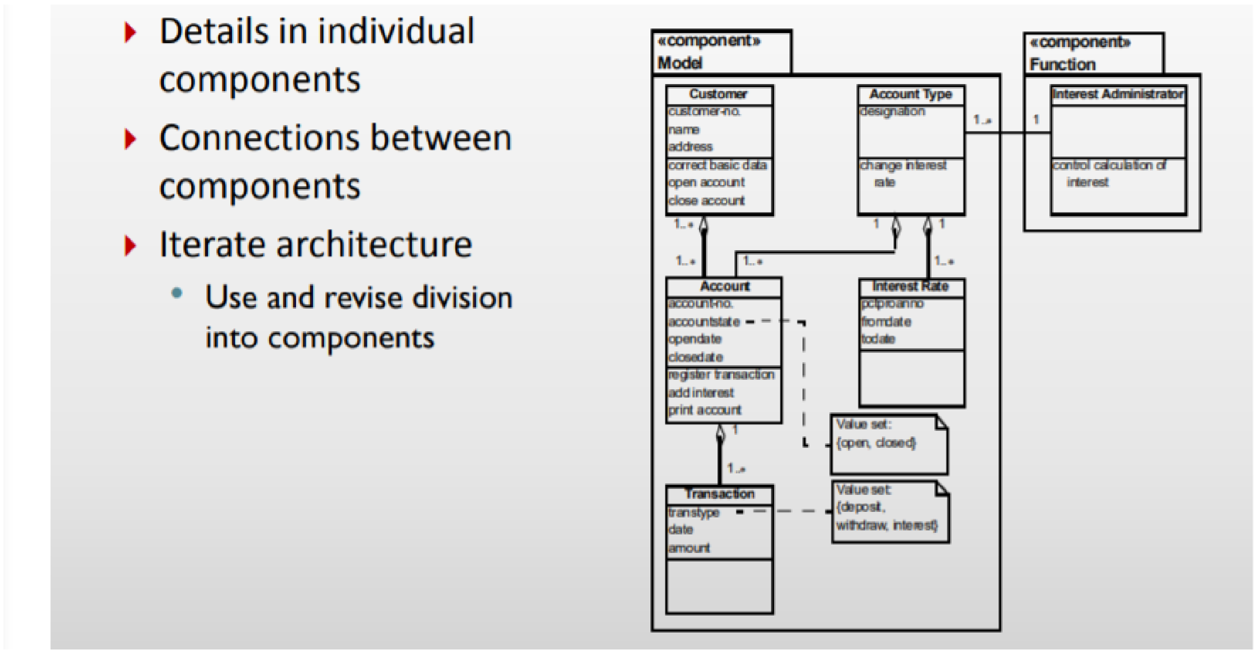
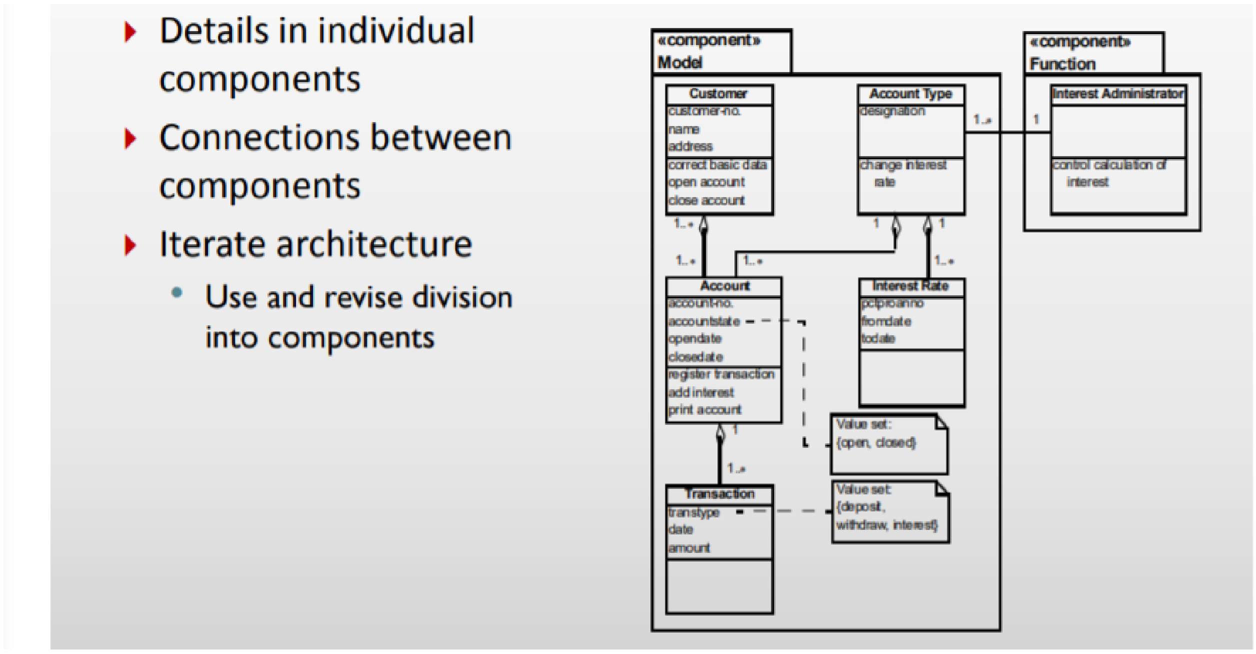
V: Components design: Putting it all together.

Results



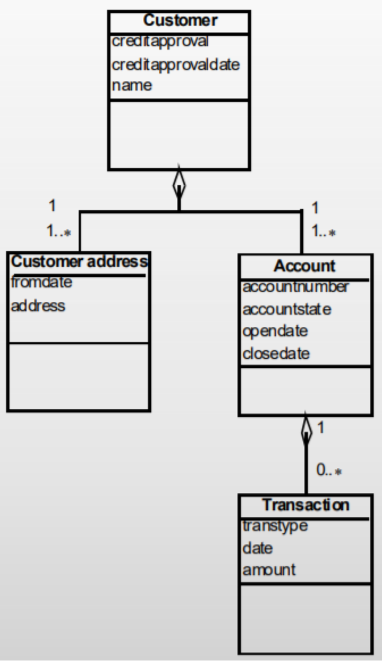
- Details in individual components
- Connections between components
- Iterate architecture
  - Use and revise division into components

Key Concepts: From Architecture to Components



- Details in individual components
- Connections between components
- Iterate architecture
  - Use and revise division into components

Model Component: Results



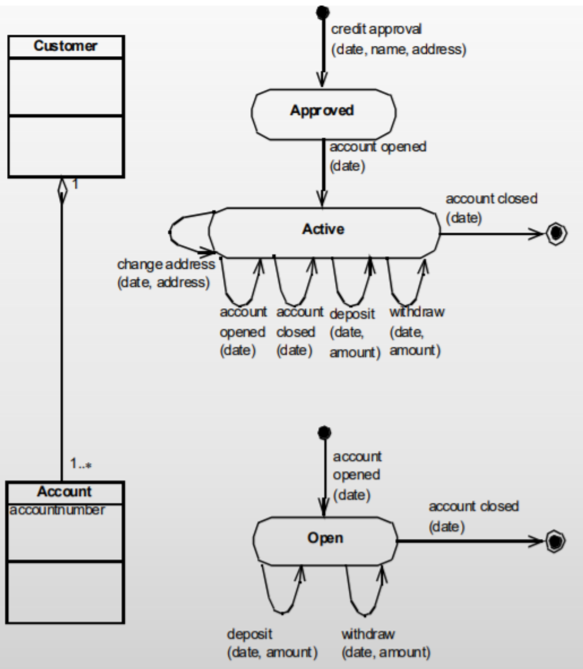
- Point of departure in the class diagram from the problem domain analysis
- Extended with representation of behavior described in the statechart diagrams

Example: bank

Analysis model:

- Class diagram
- Statechart diagrams
- Event table

Event	Customer	Account
Credit approval	+	
Change address	*	
Account opened	*	+
Account closed	*	+
Deposit	*	*
Withdraw	*	*



Represent Private Events

- Sequence and selection
    - Represent these events as a state attribute in the class described by the statechart diagram.
    - Every time one of the involved events occurs, the system shall assign a new value to the state attribute.
    - Integrate the attributes of the involved events into the class.
- Iteration
    - Represent these events as a new class; attach it to the class described by the statechart diagram using an aggregation structure.
    - For each iteration that occurs, the system shall generate a new object from the class.

    - Integrate the event attributes into the new class.

Text to: Model Component: Results: picture

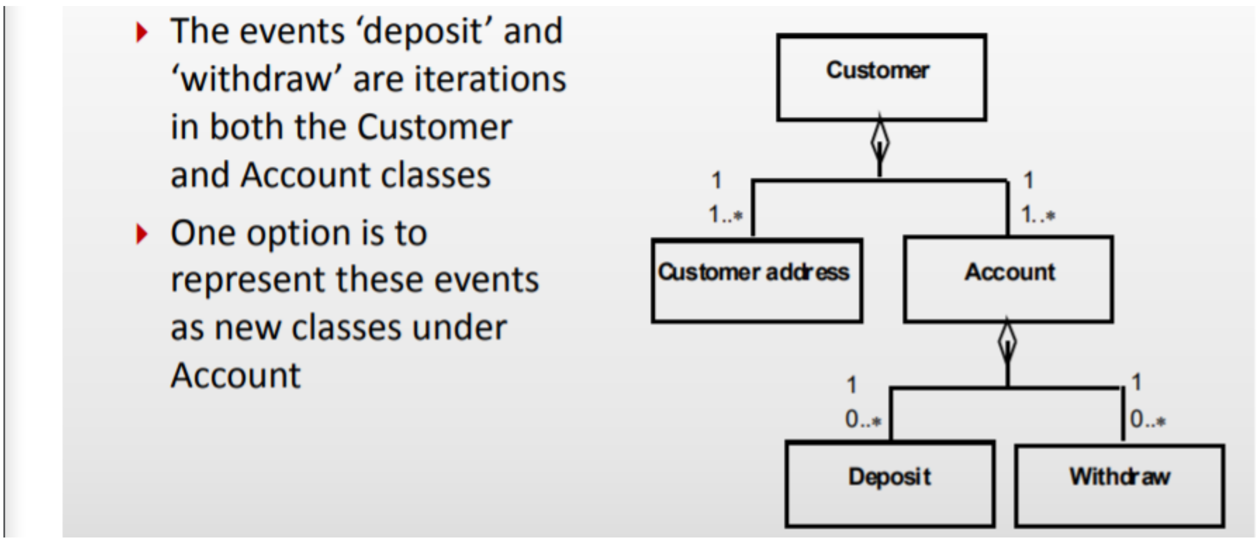
- The event 'change address' is private to the class Customer. It is an iteration in the statechart diagram of the class.
- Represent this event as a new class
- The event 'credit approval' is private to the class Customer. It is part of a sequence in the statechart diagram
- Represent this event as an attribute

Represent Common Events

Common events:

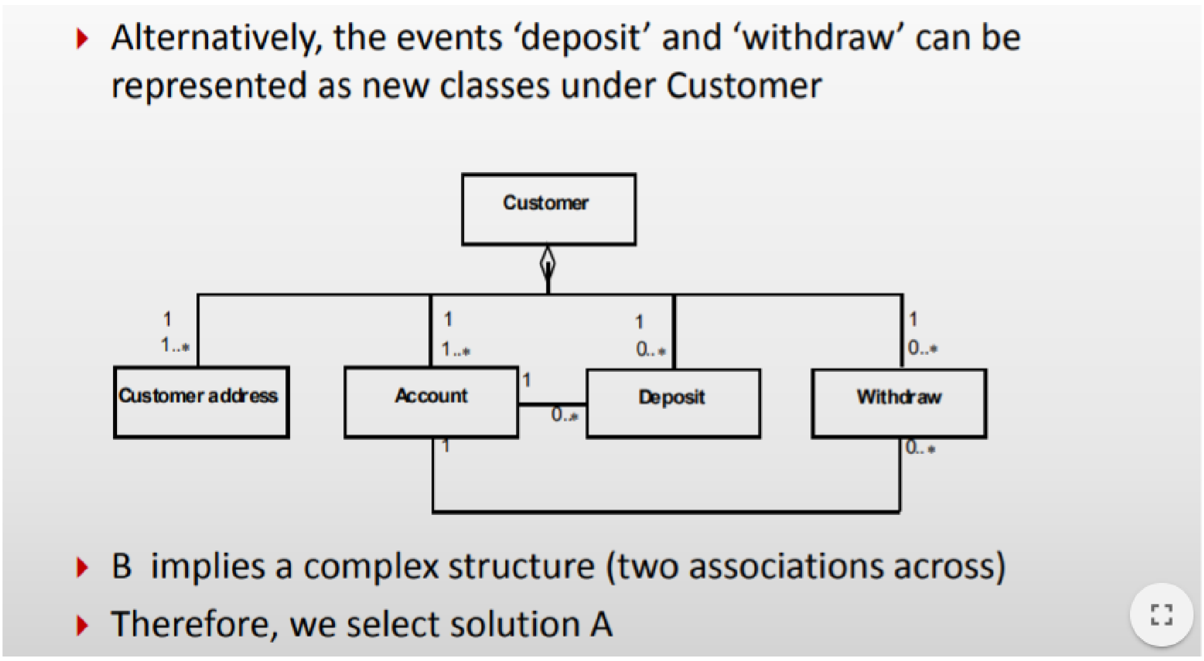
- If the event is involved in the statechart diagrams in different ways, represent it in relation to the class that offers the simplest representation.
- If the event is involved in the statechart diagrams in the same way, you must weigh possible representations against each other

Represent Common Events: Solution A



- The events 'deposit' and 'withdraw' are iterations in both the Customer and Account classes
- One option is to represent these events as new classes under Account

Represent Common Events: Solution B



- Alternatively, the events 'deposit' and 'withdraw' can be represented as new classes under Customer

- B implies a complex structure (two associations across)
- Therefore, we select solution A

Restructure Classes

- The revised class diagram represents the same information as the statechart diagrams
- The class diagram can often be simplified without loss of information:
  - Generalization
  - Association
  - Embedded iterations

Take case of unnecessary association!