# Modeling - Behaviour

This is the last activity in the problem domain analysis

The classes and methods are required to find the behavioral patterns we need.

We previously went over the elevator example when trying to construct an event table.

The activities consist of:

* What are the dynamic properties of objects?
* Event trace, behavioral pattern, and attribute.
  + Event tracing is a sequence of events involving a specific object.
  + A behavioral pattern is a description of possible event traces for all objects in a class.
  + Then we find the attributes, the descriptive properties of a class or an event

The subactivities:

1. **Describing behavioral patterns** - event trace, generate general patterns, study common events (events that several objects take part in)  
   We also try to make our behavioral models show the legal behaviour of an object, because then we also show what behaviour is ILLEGAL.
2. **Explore patterns** - We’ll be looking at three different patterns - the stepwise relation pattern, the stepwise role pattern, and composite pattern.
3. **Consider structures and classes** - Study structure and behaviour, remember that behaviour can be inherited.
4. **Describe attributes -** Derive class attributes from behavioral patterns.

If we have a bank account, a general thing we always look for are the events that create our objects and the events that terminate it. We always have to find out when the object starts existing in our system and ends. In a bank system, a customer starts existing when they open an account. Based on that event, we get a *state* of the object. Look at slide 8 for a diagram describing the state of the customer. Amounts can be deposited and withdrawn, but those events don’t change the event. This is an unstructured state chart diagram.

Slide 9 is a more structured state chart diagram. This is an example of an author that starts existing when a paper is registered. Before anything else can happen, the paper must be submitted. After submitting, there are no more events that involve the author.

The main difference between the structured and unstructured diagram is that the structured diagram is a sequence, where the unstructured one is only sequentialized by two very special events, and the rest do not change the state of the object or come before or after each other.

The reason behind this distinction is that structured behavioral patterns are very important, and if we have an unstructured behavioral pattern, we need to look for structures in it.

Look at slide 10 for statechart diagram notation. It is rather self explanatory.

Sometimes, an object can toggle between two states, which is shown in the bottom right corner of slide 10. This is called “indirect iteration”.

## Find behavioral patterns from event traces

For each class:

* Which events cause the creation of a problem domain object?
* Which events cause the death of a problem domain object?
  + When an object is “dead”, it just means that it cannot be involved in any more events.

Typical events traces

* Which events occur in a sequence?
* Is the overall form structured or unstructured?
* Are there any alternative events?
* Can a given event occur more than once?

### Example

We return to the hair saloon problem. A customer starts existing in our problem domain when they first make an appointment. After this, they can cancel reservations, make new reservations, and get treated. During this, they are considered active. We have also identified three attributes; name, address, and phone number.

The appointment object starts existing when a reservation is made by a customer. After activation, the state of the appointment is “planned”. The appointment can be cancelled or it can be “treated”, both leading out of the “planned” state.

## Hierarchical states

This is where you have multiple events and states within an overall state. For instance, if we have identified the sequence of events that leads to different states, but during each of these states, the object can be terminated, then we have to option of creating an overall state and let the termination arrows go from the overall state to the termination state. See slide 14 for an example. It’s the bottom example, which is easier to read than the top one.

Look at the apprentice statechart. An apprentice starts existing when they are employed, and during this time, they can always be resigned, and thereby terminated.

## Inheritance of behavioral patterns

When we have different specializations of a superclass, we can identify behavioral patterns that can be inherited by all subclasses. See slide 16. The superclass has a behavioral pattern that is inherited by the subclass, but is also extended with another behavioral pattern. Also, see slide 18 for another example of behavior inheritance.

## Stepwise relation pattern

An object cannot enter a relation to objects of another class until after another relation of higher hierarchy is already established. The stepwise relation pattern is only concerned with objects in a hierarchy. For example, a student must be assigned to a semester before being assigned to a class. There is a specific sequence in which the relations can be made.

## Stepwise role pattern

Remember how we modeled roles when objects could dynamically change between different roles. In this pattern, we have a defined sequence in which the different roles can be entered. For example, a sale can be either an offer, and order, or a delivery. However, these roles must happen in a specific sequence.

## Composite pattern

Useful for describing the creation or destruction of a hierarchical structure that is unknown at the time of model development. The general abstraction is that we have an aggregation of something like “a car engine aggregates a lot of different parts”. We don’t know the specifics. We can model it by saying a part is either a simple part, or a composite of other parts. For simple parts, once we have it, it’s ready. If it’s not, then it’s a composite, then we need to start assembling it (state: assembly), and when the part is readily put together, it can be mounted in the overall object.

In short, for different parts, they will always have a “ready” state, what matters is how and when we get to that state depending on whether the part is a simple one or a composite.

## Evaluate classes and structure

When we have the behavioral patterns, we evaluate the classes and structures. We look for possible generalizations that we didn’t see. If the same event is tied to two classes, we can consider that one class is a generalization of the other class. At object level, we can look at two objects having common events. Also, if a class aggravates another class, they should share at least one common event.

For instance, the bank customer was a little bit confusing because when an account opened, the account state was open, and the withdraw and deposit were iterative events that didn’t change the class immediately. We can change the class to help us make sense of the diagram. See slide 23 for an example.

Return to the hair saloon, and look at the plan class.

Every day, he creates a daily schedule. It starts existing when it’s scheduled. There is only one way out of the “planned” state, and that is to agree the schedule. During the “agreed” state, several iterative events can occur.

Remember that when we made event tables last, we used checkmarks. Now, we know a bit more, whether an event can happen once or multiple times. The plus signs indicate a single occurrence, and a star indicates several occurrences.

## Describe attributes

We distinguish between three types:

* Attributes connected to events
* General information about the objects in the classes (name, address)
* Attributes that we can derive from other attributes.

For classes:

* What are the general characteristics of the class?
* How is the class described in the problem domain?
* What basic data must be captured about objects from this class? This can be both static and dynamic objects. If objects are state-changing, we might need to capture information about this.
* What results from an event trace must be captured?

For events:

* What time did the event occur?
* Which amount did it concern?

## Result

The result of a behavioral pattern analysis is a behavioral pattern with attributes for every class in a class diagram.

# Exercises

5.15

Legal event trace:

* Account is created on XX/XX/XX. 100 dollars are deposited. 50 dollars are withdrawn. Account is closed.

Illegal event trace:

* Account is created on XX/XX/XX. 100 dollars are deposited. Account is closed. 50 dollars are withdrawn.

5.17

**Stepwise relation pattern**

A student must have a relation to a semester before having a relation to class in that semester. Thereafter the student can have a relation to a group in that class.

A car mechanic must have a relation to a car before generating relations with the components that are faulty. After this, he can generate a relation to the specific part that is faulty. A car aggravates components that aggravate parts.

**Stepwise role pattern**

**Noter:**

Attributter skal IKKE referere til objekter.