Esper-Praktikum

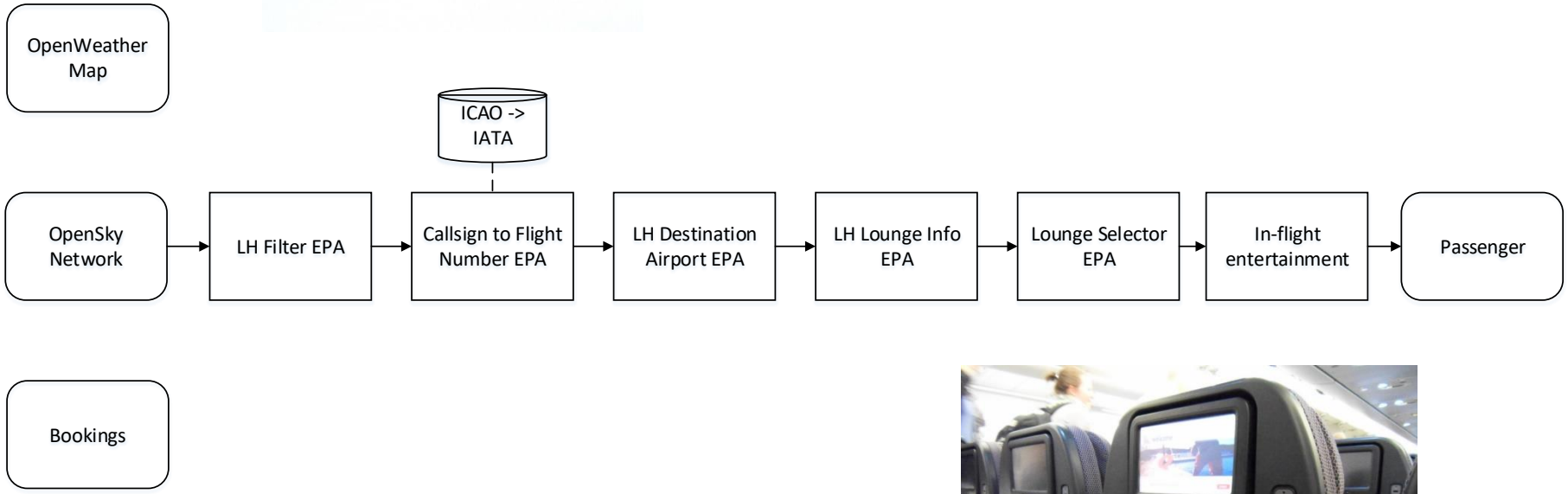
Complex Event Processing

Sommersemester 2019

Fachgebiet Business Process Technology
Stephan Haarmann

(Original: Kimon Batoulis, Sanklaita Mandal)

Use Case: Airline EPN



Wetter-Ereignis London

OpenWeather Map


```
{
    "coord": {
        "lon": -0.13,
        "lat": 51.51
    },
    "weather": [
        {
            "id": 300,
            "main": "Drizzle",
            "description": "light intensity drizzle",
            "icon": "09d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 280.32,
        "pressure": 1012,
        "humidity": 81,
        "temp_min": 279.15,
        "temp_max": 281.15
    },
    "visibility": 10000,
    "wind": {
        "speed": 4.1,
        "deg": 80
    },
    "clouds": {
        "all": 90
    },
    "dt": 1485789600,
    "sys": {
        "type": 1,
        "id": 5091,
        "message": 0.0103,
        "country": "GB",
        "sunrise": 1485762037,
        "sunset": 1485794875
    },
    "id": 2643743,
    "name": "London"
```

Ereignistyp StateVector

OpenSky Network

Index	Property	Type	Description
0	*icao24*	string	Unique ICAO 24-bit address of the transponder in hex string representation.
1	*callsign*	string	Callsign of the vehicle (8 chars). Can be null if no callsign has been received.
2	*origin_country*	string	Country name inferred from the ICAO 24-bit address.
3	*time_position*	int	Unix timestamp (seconds) for the last position update. Can be null if no position report was received by OpenSky within the past 15s.
4	*last_contact*	int	Unix timestamp (seconds) for the last update in general. This field is updated for any new, valid message received from the transponder.
5	*longitude*	float	WGS-84 longitude in decimal degrees. Can be null.
6	*latitude*	float	WGS-84 latitude in decimal degrees. Can be null.
7	*geo_altitude*	float	Geometric altitude in meters. Can be null.
8	*on_ground*	boolean	Boolean value which indicates if the position was retrieved from a surface position report.
9	*velocity*	float	Velocity over ground in m/s. Can be null.
10	*heading*	float	Heading in decimal degrees clockwise from north (i.e. north=0°). Can be null.
11	*vertical_rate*	float	Vertical rate in m/s. A positive value indicates that the airplane is climbing, a negative value indicates that it descends. Can be null.
12	*sensors*	int[]	IDs of the receivers which contributed to this state vector. Is null if no filtering for sensor was used in the request.
13	*baro_altitude*	float	Barometric altitude in meters. Can be null.
14	*squawk*	string	The transponder code aka Squawk. Can be null.
15	*spi*	boolean	Whether flight status indicates special purpose indicator.
16	*position_source*	int	Origin of this state's position: 0 = ADS-B, 1 = ASTERIX, 2 = MLAT

Ereignistyp Booking

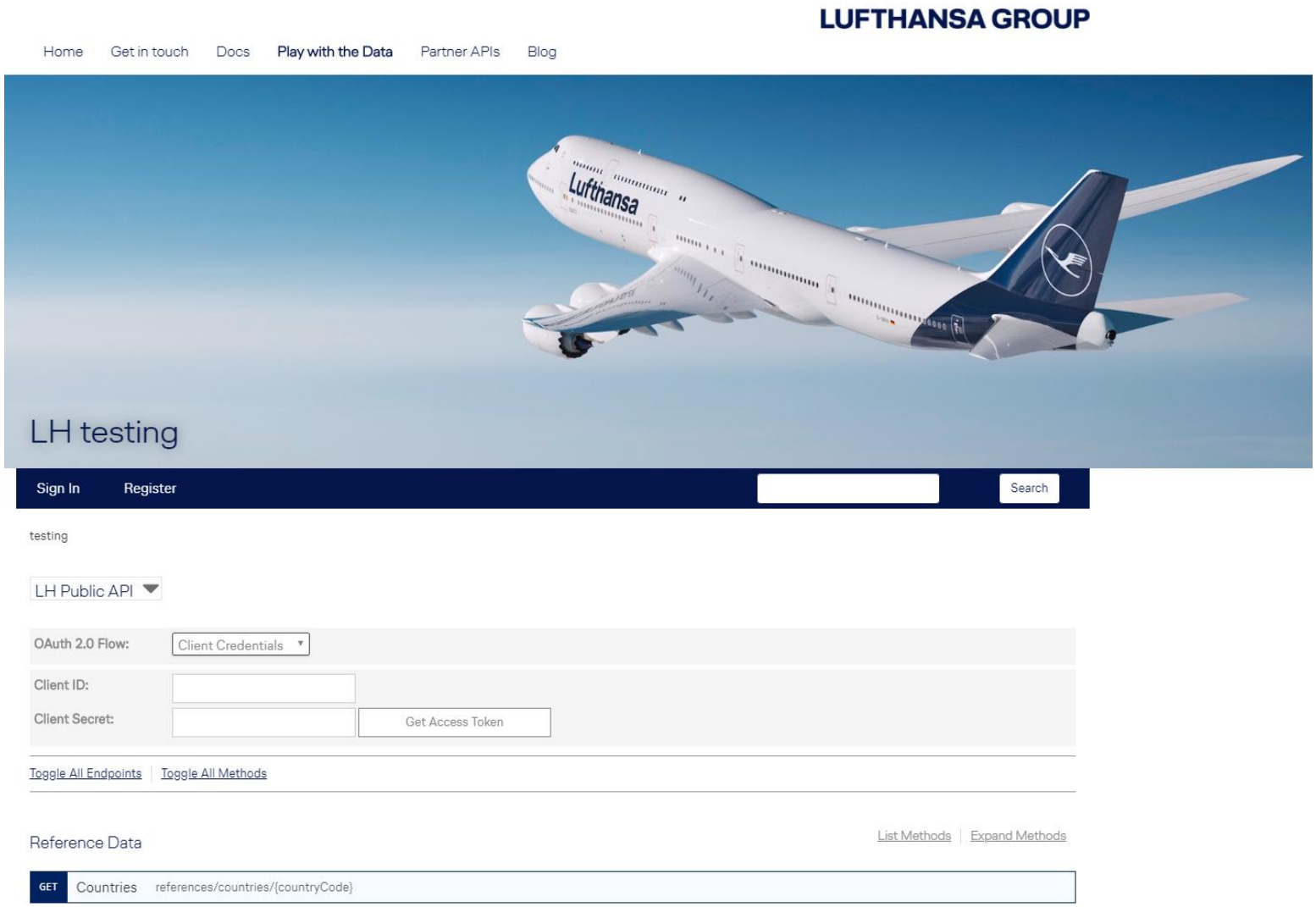

```
public class Booking {
    private String flightNumber;
    private CabinClass cabinClass;
    private String passengerName;


public enum CabinClass { ECONOMY, PREMIUM_ECONOMY, BUSINESS, FIRST }
```

Ereignistyp StateVector

OpenSky Network

Index	Property	Type	Description
0	*icao24*	string	Unique ICAO 24-bit address of the transponder in hex string representation.
1	*callsign*	string	Callsign of the vehicle (8 chars). Can be null if no callsign has been received.
2	*origin_country*	string	Country name inferred from the ICAO 24-bit address.
3	*time_position*	int	Unix timestamp (seconds) for the last position update. Can be null if no position report was received by OpenSky within the past 15s.
4	*last_contact*	int	Unix timestamp (seconds) for the last update in general. This field is updated for any new, valid message received from the transponder.
5	*longitude*	float	WGS-84 longitude in decimal degrees. Can be null.
6	*latitude*	float	WGS-84 latitude in decimal degrees. Can be null.
7	*geo_altitude*	float	Geometric altitude in meters. Can be null.
8	*on_ground*	boolean	Boolean value which indicates if the position was retrieved from a surface position report.
9	*velocity*	float	Velocity over ground in m/s. Can be null.
10	*heading*	float	Heading in decimal degrees clockwise from north (i.e. north=0°). Can be null.
11	*vertical_rate*	float	Vertical rate in m/s. A positive value indicates that the airplane is climbing, a negative value indicates that it descends. Can be null.
12	*sensors*	int[]	IDs of the receivers which contributed to this state vector. Is null if no filtering for sensor was used in the request.
13	*baro_altitude*	float	Barometric altitude in meters. Can be null.
14	*squawk*	string	The transponder code aka Squawk. Can be null.
15	*spi*	boolean	Whether flight status indicates special purpose indicator.
16	*position_source*	int	Origin of this state's position: 0 = ADS-B, 1 = ASTERIX, 2 = MLAT

Lufthansa API



GET Flight Status operations/flightstatus/{flightNumber}/{date}

Status of a particular flight (boarding, delayed, etc.).

Parameter	Value	Type	Description
{flightNumber}	LH123	string	Flight number including carrier code and any suffix (e.g. 'LH400')
{date}	2018-06-07	date	The departure date (YYYY-MM-DD) in the local time of the departure airport
limit		string	Number of records returned per request. Defaults to 20, maximum is 100 (if a value bigger than 100 is given, 100 will be taken)
offset		string	Number of records skipped. Defaults to 0
Accept	application/json	string	http header: application/json or application/xml

Try it! Clear Results

Request URI

```
https://api.lufthansa.com/v1/operations/flightstatus/LH123/2018-06-07
```

Request Headers Select content

```
Accept: application/json
Authorization: Bearer zdbmqqnynjvk3qe38gxa4kw7
X-Originating-IP: 141.89.226.146
```

Response Status Select content

```
200 OK
```

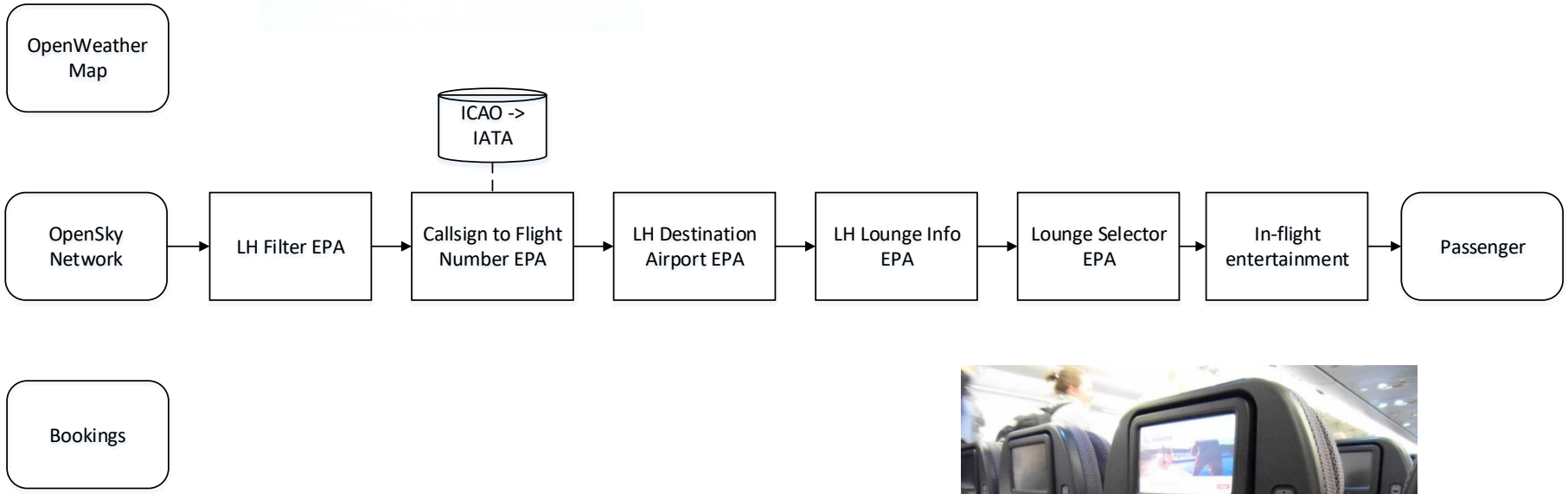
Response Headers Select content

```
Content-Type: application/json; charset=UTF-8
Date: Thu, 07 Jun 2018 08:48:08 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
X-Mashery-Message-Id: 0ba736d0-caf2-4ecd-bc17-fee87cd7d6e3
X-Mashery-Responder: prod-j-worker-eu-west-1c-23.mashery.com
Content-Length: 974
Connection: keep-alive
```

Response Body Select content

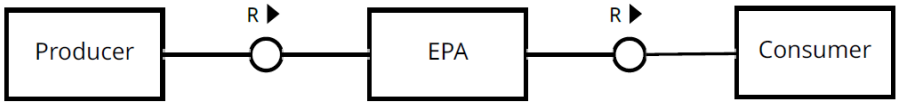

```
{
    "FlightStatusResource": {
        "Flights": {
            "Flight": {
                "Departure": {
                    "AirportCode": "MUC",
                    "ScheduledTimeLocal": {
                        "DateTime": "2018-06-07T21:30"
                    },
                    "ScheduledTimeUTC": {
                        "DateTime": "2018-06-07T19:30Z"
                    },
                    "TimeStatus": {
                        "Code": "OT",
                        "Definition": "Flight On Time"
```

Use Case: Airline EPN

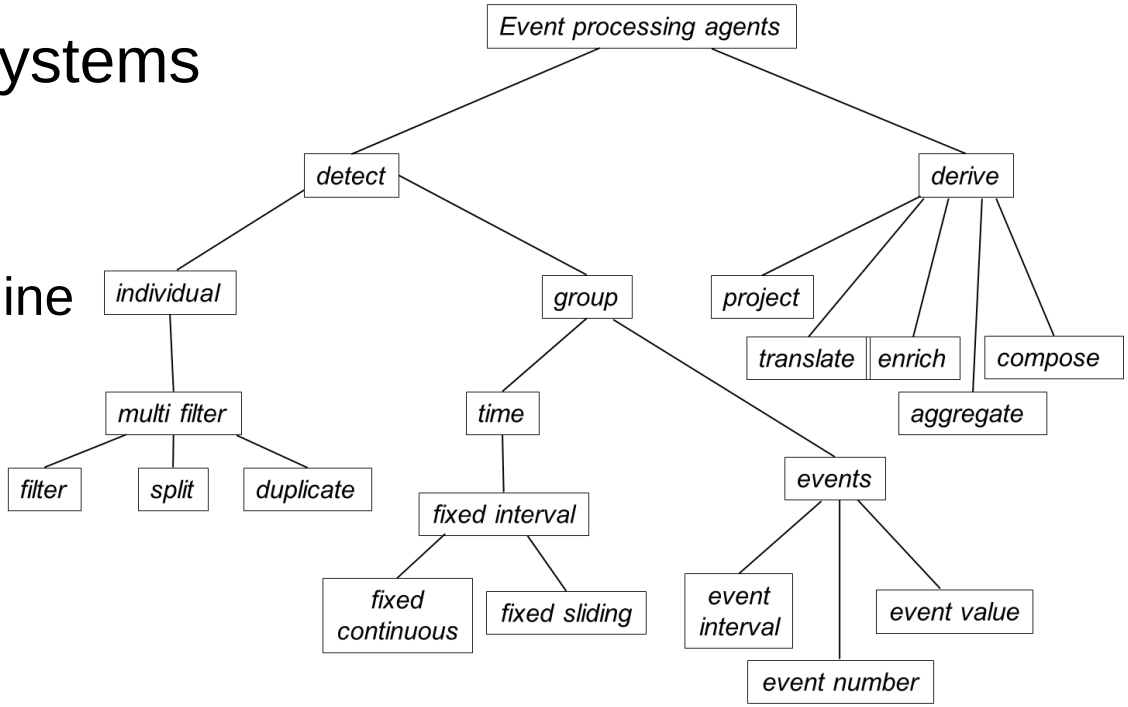


Rückblick

1. Einführung
2. Grundlagen der Ereignisverarbeitung
   1. Auftreten und Erkennen von Ereignissen
   2. Event Processing Networks

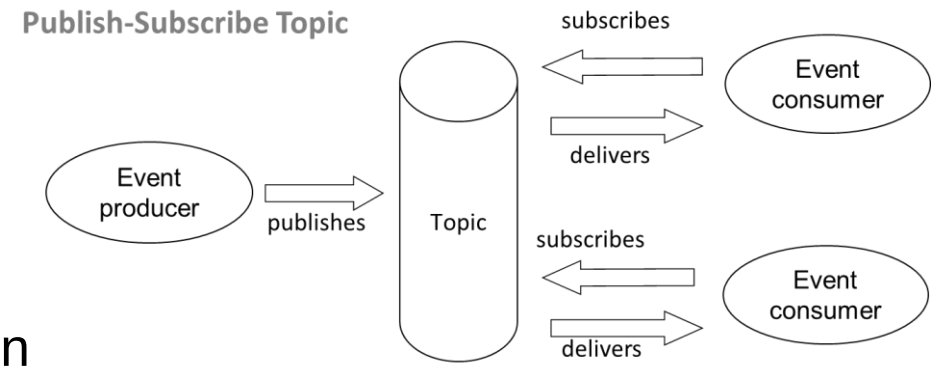


3. Event Processing Systems
   1. Ereigniserkennung
   2. Ereigniserzeugung
   3. Beispielszenario Airline

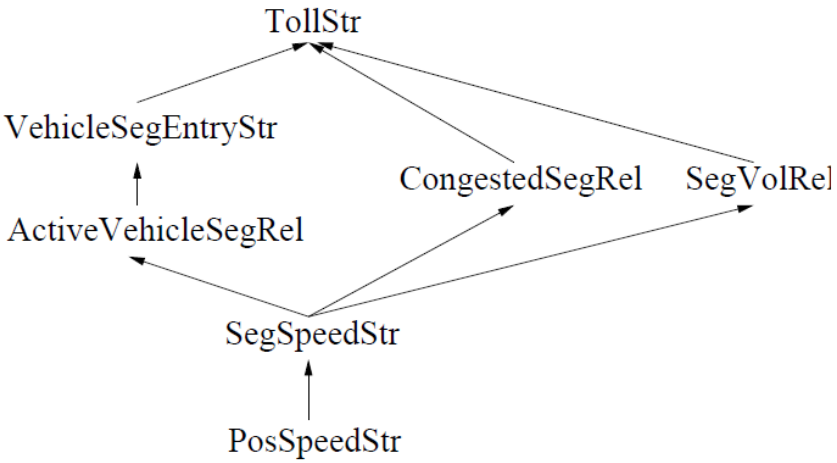


Rückblick

4. Kommunikationsmodelle
   1. Direkte Kommunikation
   2. Kanalkommunikation
   3. Kommunikationsverbindungen



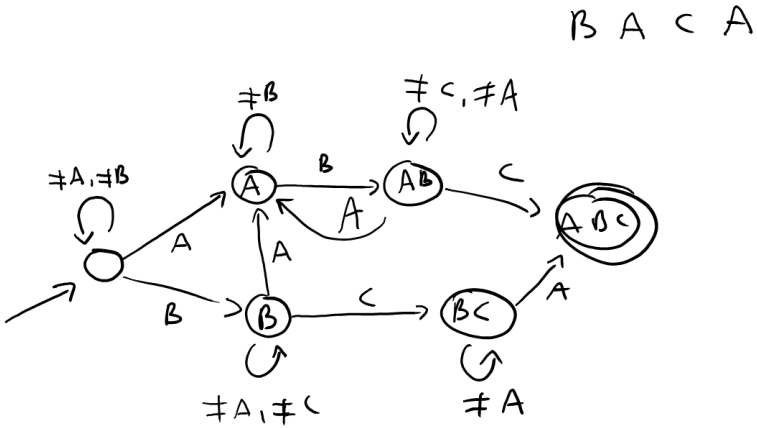
5. Continuous Query Language
   1. Grundlagen von CQL
   2. Operatoren der CQL
   3. Zusammenfassendes Beispiel



6. Esper EPL
   1. Einführung
   2. Umsetzung von EPA in Esper

Rückblick

7. Mustererkennung
   1. Wiederholungsmuster
   2. Mustererkennung durch Automaten
   3. Optimierte Mustererkennung
   4. Umsetzung in Esper EPL



8. Ereignisverarbeitung im Prozessmanagement
   1. Ereignisse in der BPMN
   2. Publish / Subscribe für Prozessereignisse

9. Praktische Umsetzung
   1. Unicorn und Chimera
   2. Ereignisse und Prozesse
   3. Beispiel und Praktikum