



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

Semesterproject

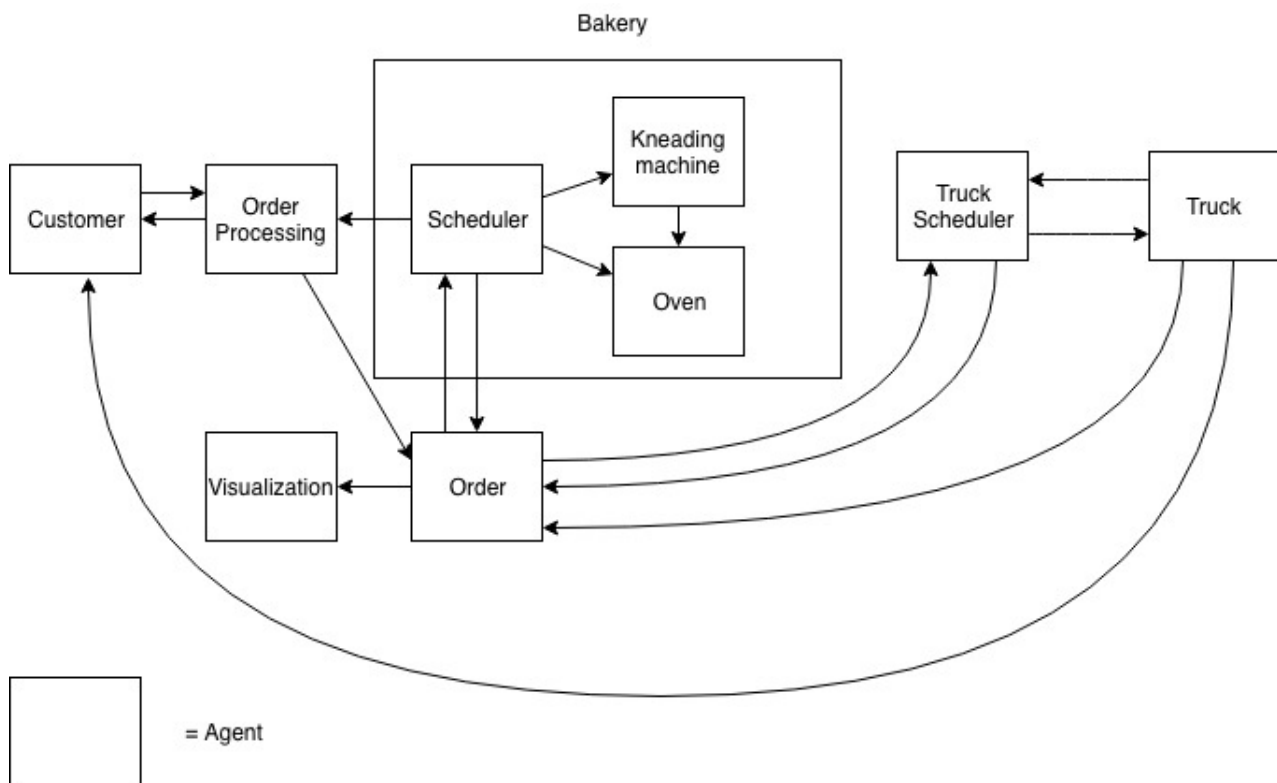
Multi Agent and Agent Systems

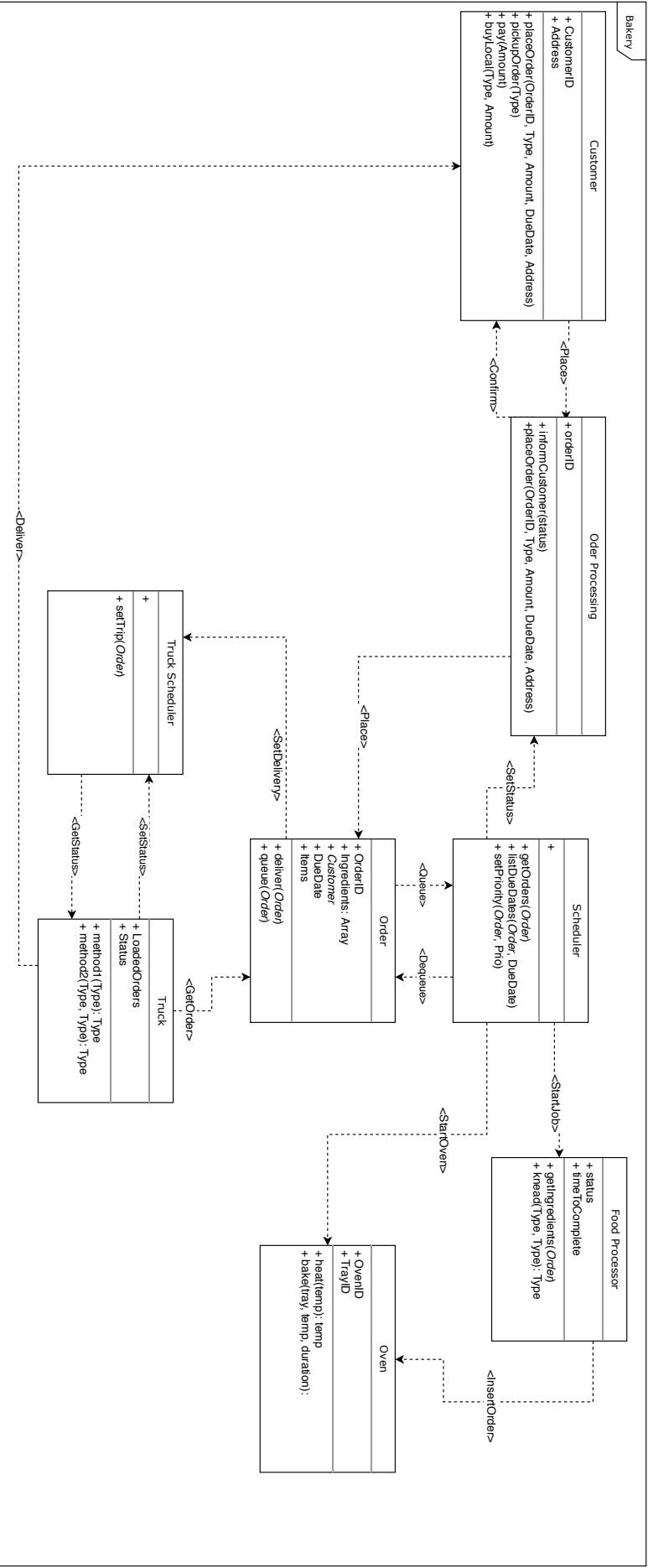
Department of Autonomous Systems
Referent: Prof. Dr. Gerhard Kraetzschmar

Submitted by: Team PJT

Sankt Augustin 4. November 2018

Architecturepicture





Aggregation of order data

Aggregation of order data can be done in the following manner:

- **An aggregation of a customer's orders for each day or each date <ddd.hh>**
→ Use of a Hashmap. Key is date value is order. The advantage is that a hashmap has got an index. That means that worst case runtime for searching for an order within hashmap is $O(n) = 1$

```
HashMap<Date, Order> hmMapDaily = new HashMap<Date, Order>();  
hmMapDaily.put(new Date(), new Order());  
Order co = hmMapDaily.get(date);
```

- **An aggregation of all orders for a particular product for each day or each date**
→ Hashmap of Hashmaps. One entry within Hashmap represents one product. Key is product value is a hashmap. One Hashmap within Hashmap has as key a date, as value an array of orders.

```
HashMap<ProductId, HashMap<Date, Orders[]>> hMapProduct;  
hMapProduct.put(new ProductId(), HashMap<Date, Orders[]>());  
HashMap<Date, Orders[]> hmDate = hMapProduct.get(ProductId);
```

So hMapProduct would look the following way:

$$hMapProduct = \begin{pmatrix} \{ProductId, HashMap < Date, Orders[] >\} \\ . \\ . \\ . \\ \{ProductId, HashMap < Date, Orders[] >\} \end{pmatrix}$$