Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Semesterproject

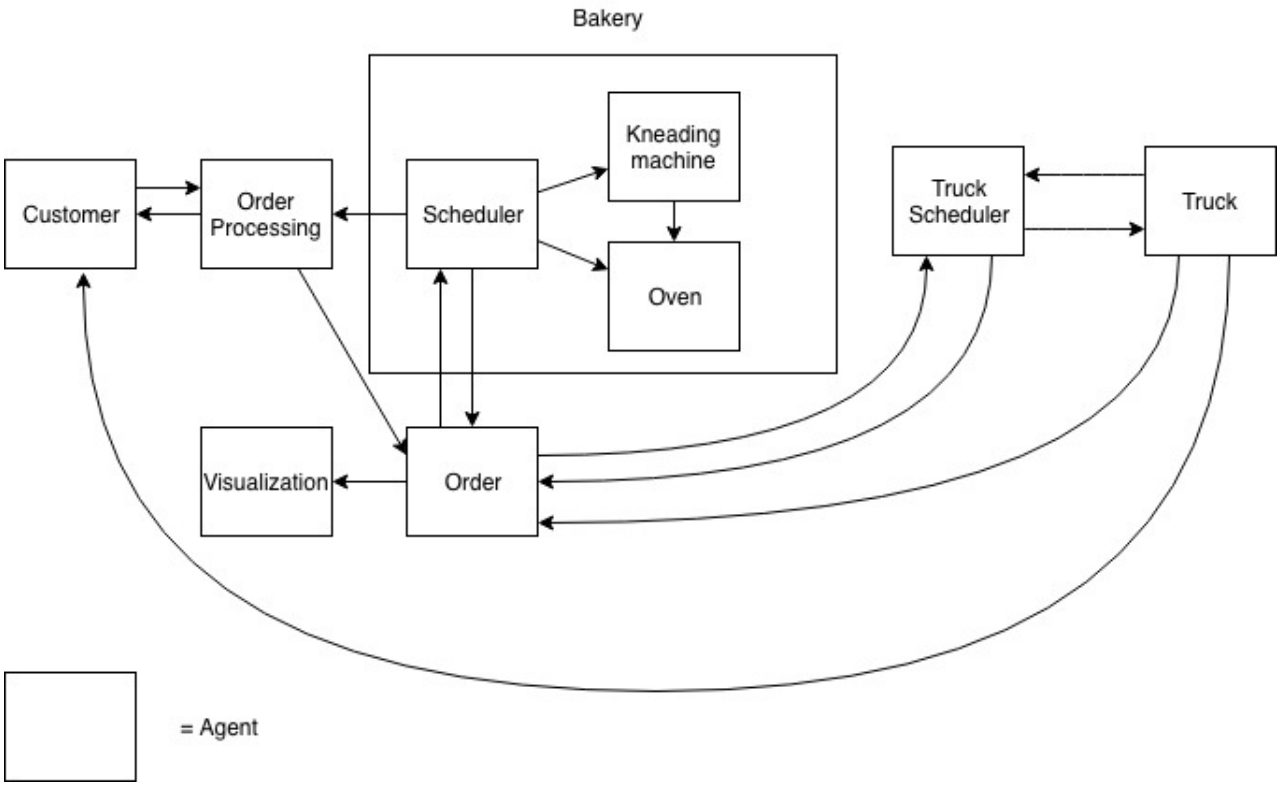
Multi Agent and Agent Systems

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

Submitted by:
team pjt

Sankt Augustin 07.11.2018

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

  So hmMapProduct would look the following way:

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