Hochschule
Bonn-Rhein-Sieg

Amazon Warehouse / Kiva

Patrick Robinson

Bastian Mager

Daniel Pyka

Hochschule Bonn-Rhein-Sieg

Content

- General
- Communication
  - Order assignment
  - Order processing
- Orderpicker
  - Internal data structure
  - Control flow
- Monitoring
- Configuration
- Scale-up the system

General

- Data transferal: JSON
- Action differentiation: JADE Performatives
- Configurability: JADE or JSON
- Customer simulation:
  - Ticking: Repeatedly spawns random orders
  - Parse: Takes orders as parameters
    - Part of json config
    - Entered in JADE on customer agent creation

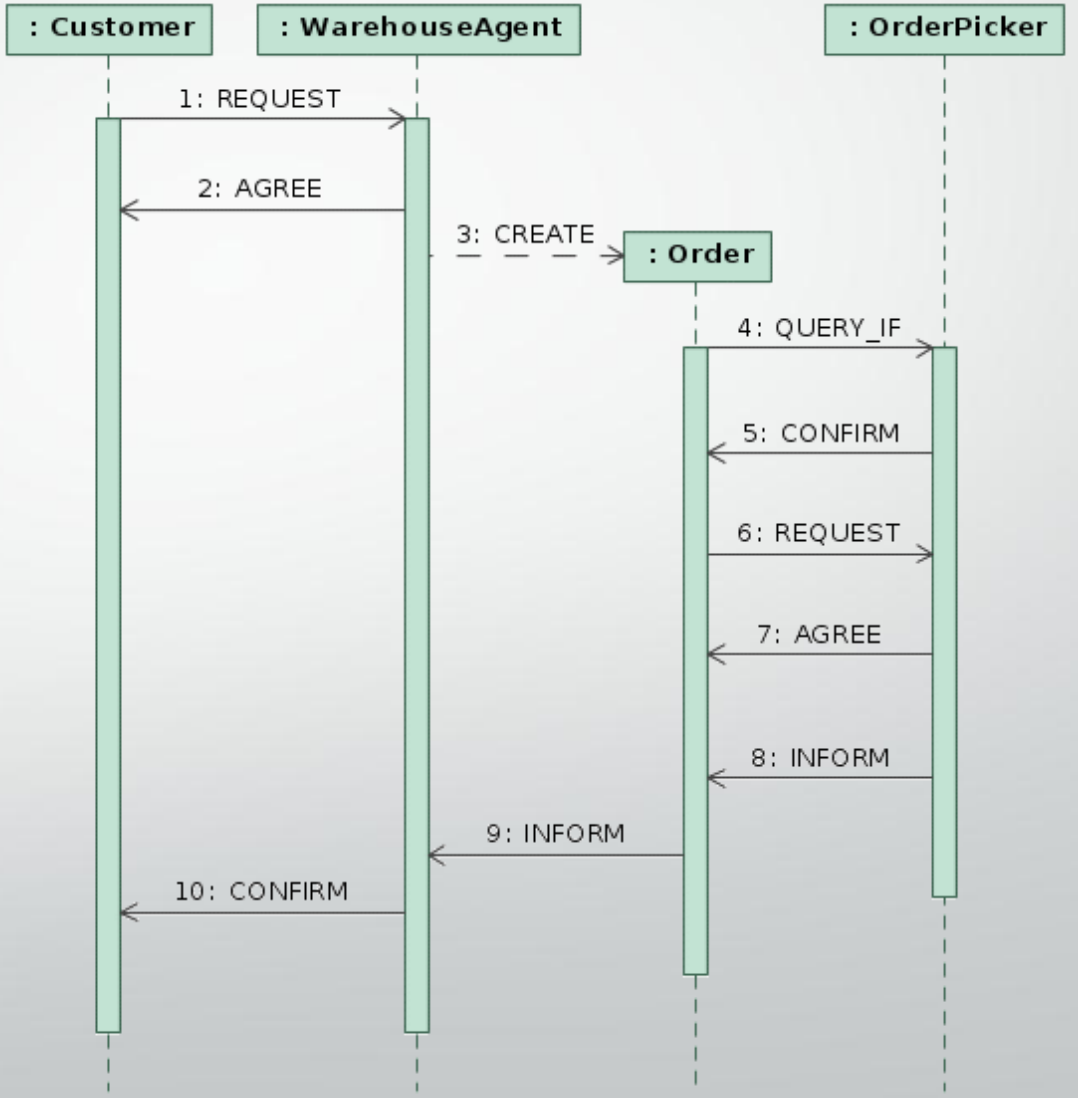
Hochschule Bonn-Rhein-Sieg

Communication

Order assignment

Data flow:

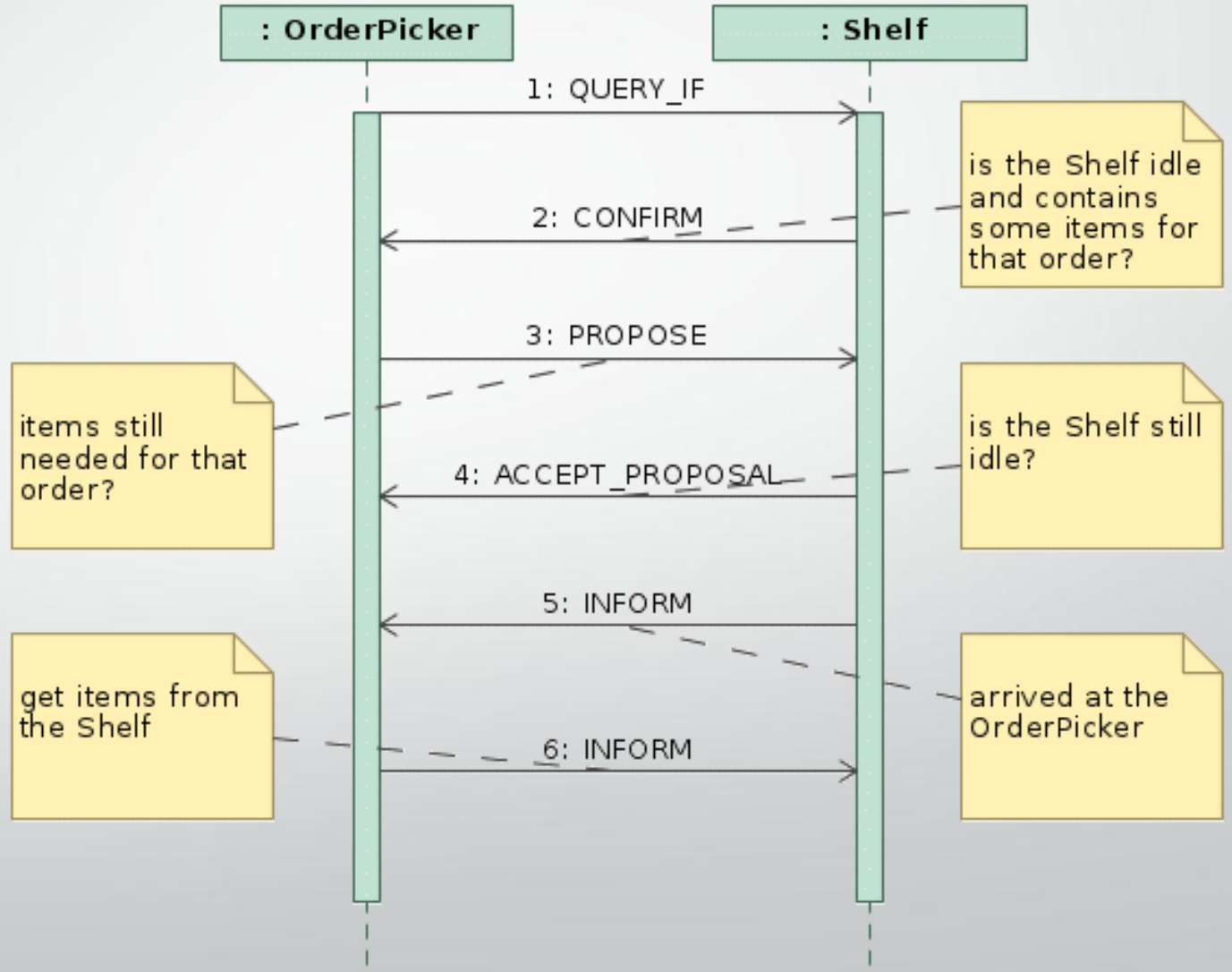
- REQUEST: {id:o,products:[{name:count}]}



Hochschule Bonn-Rhein-Sieg

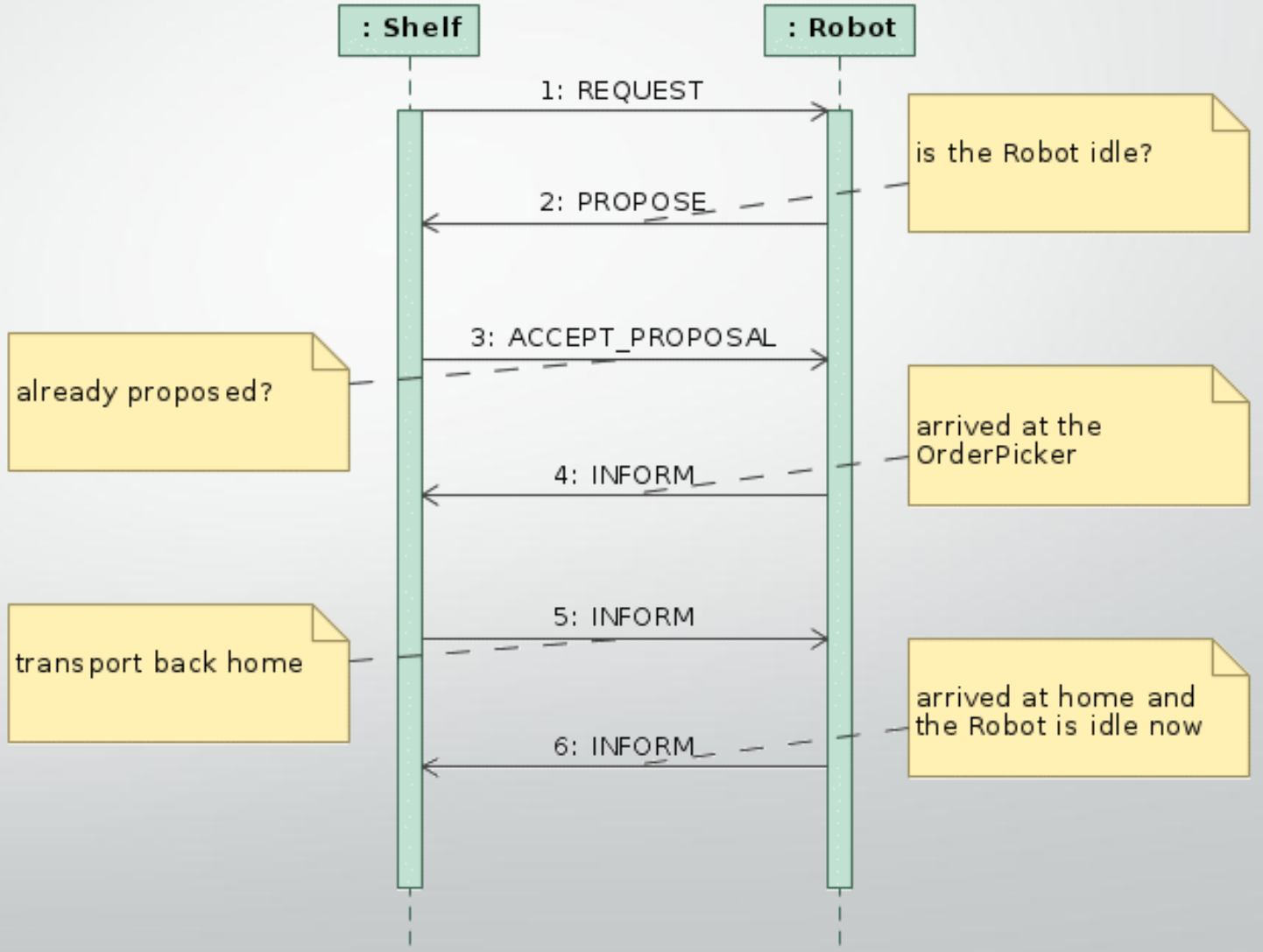
Communication

Order processing



Communication

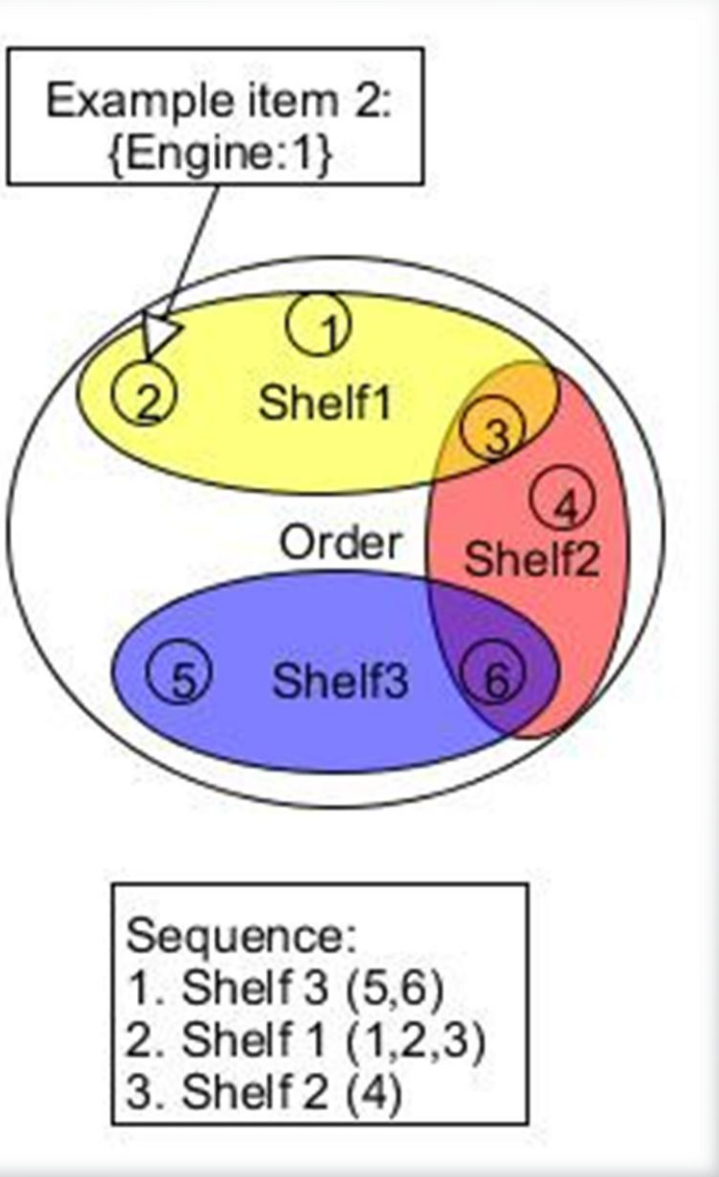
Order processing



Orderpicker

Internal data structure

- Item status enumeration: broadcasted, shelf_proposed, shelf_accepted, property
- Hashmap <item, item status>
- Hashmap <item, shelf>
- No fixed order



Orderpicker

Control flow

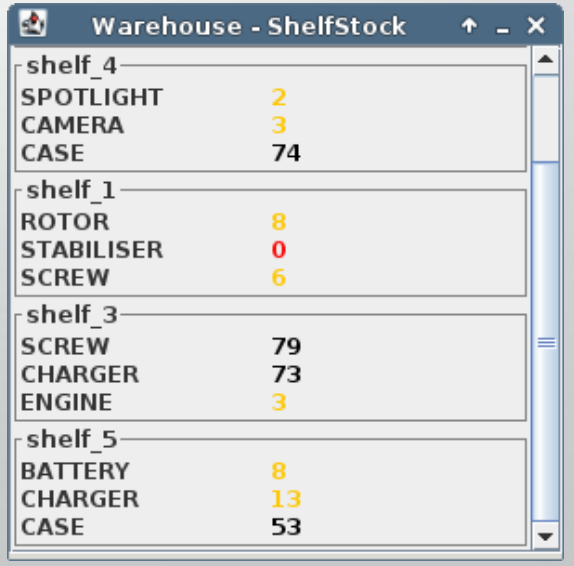
- Processing the order can fail
  - No shelfs instantiated, all shelfs busy, shelfs are empty, “reject proposal”
- Broadcast missing items again (ticker behaviour)
- Immediately rebroadcast items after receiving “reject proposal”
- Abort the order after several broadcasts (timeout)
- Return incomplete order to OrderAgent (ACLMessage.FAILURE)
- Failed order reported to CustomerAgent

Monitoring

- General Warehouse logs to console
- OrderPickers generate logfiles for every Order due to communication complexity and better traceability
- Shelfs update their stocks „live“ in a window

orderpicker2_Order1000.txt

```
316
317   ---shelfInteraction---
318   from: shelf_4
319   aclmessage: INFORM
320   content: [{"CAMERA":9},{"SPOTLIGHT":5}]
321
322   ---itemStatus---
323   {BATTERY:2} : PROPERTY
324   {CAMERA:9} : PROPERTY
325   {ROTOR:6} : PROPERTY
326   {CHARGER:2} : PROPERTY
327   {CASE:7} : PROPERTY
328   {ENGINE:1} : PROPERTY
329   {STABILISER:8} : PROPERTY
330   {CIRCUIT:4} : PROPERTY
331   {SPOTLIGHT:5} : PROPERTY
332   {TUNER:1} : PROPERTY
333   {SCREW:8} : PROPERTY
334
335   Order complete, send INFORM to Order1000!
336
337
```



Configuration

- Define kiva.config.json
- Execute parse.js to get JADE call

```
{
    shelves: [ {uid:0, products:[
        {name:"x",stock:{current:10}},
        {name:"y",stock:{current:10}}
    ]}],
    pickers: [{uid:0}, {uid:1}],
    robots: [{uid:0}, {uid:1}],
    orders: [{uid:0, products:[
        {name:"x", quantity:1},
        {name:"y", quantity:3}
    ]}]
}
```

Scale-up the system/ limits

- Scenario: 3 new orders per second, 10 orderpickers, 20 shelfs. How many robots are at least required to process each order without timeout?
  - => You can not say for sure, estimate
- Several variables (number of orderpickers, number of rebroadcasts, interval, robot travel delay, item distribution among shelfs, message queue)
- Number of robots too low: Very slow order processing? -> No order processing most likely (timeout)
- Unless you chose “extreme” values for instantiating agents, the system will work