Introduction to Multi Agent Systems

Project: Dutch Auction Agent

17.12.2014

Frank Wittich
Lucas Lersch

Agenda

- ➔ Dutch Auction
- ➔ Agent Features
- ➔ Application Structure
- ➔ Implementation Features
- ➔ Demonstration
- ➔ Improvements
- ➔ Conclusion

Reminder: Dutch Auction

Dutch Auction

- ➔ First come first serve principle
- ➔ Normally high starting price → steadily decreasing
- ➔ In our case reversed → increasing from low starting price
- ➔ Optimal Strategy: Bid what you are willing to pay

Environment

- ➔ Partially observable
- ➔ Discrete
- ➔ Multi-agent
- ➔ Stochastic

Formal Agent Features

Agent Architecture

➔ Reactive

Auction situation & behavior

➔ Individual competition over resources
➔ Risk neutral/prone

Offer evaluation

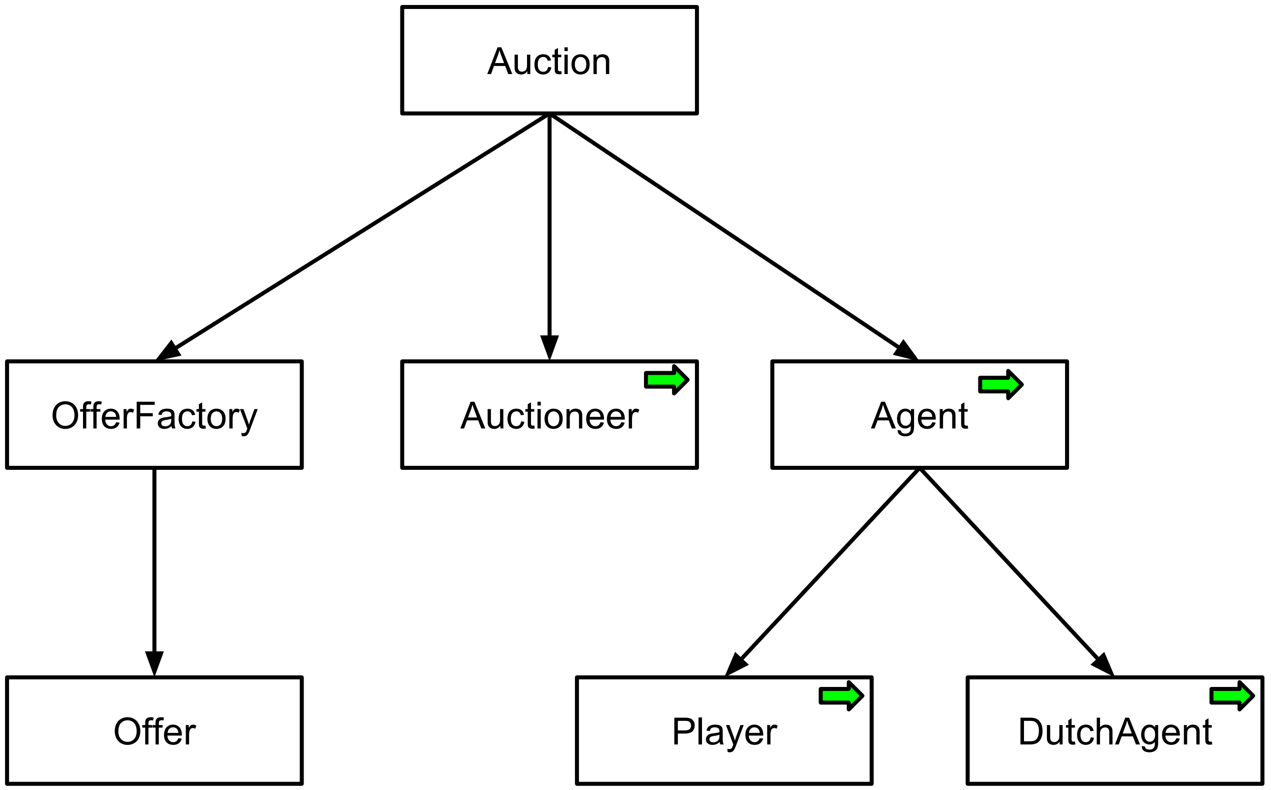
➔ Correlated value

Learning

➔ Examples and practice
➔ Unsupervised learning (trial-and-error principle)

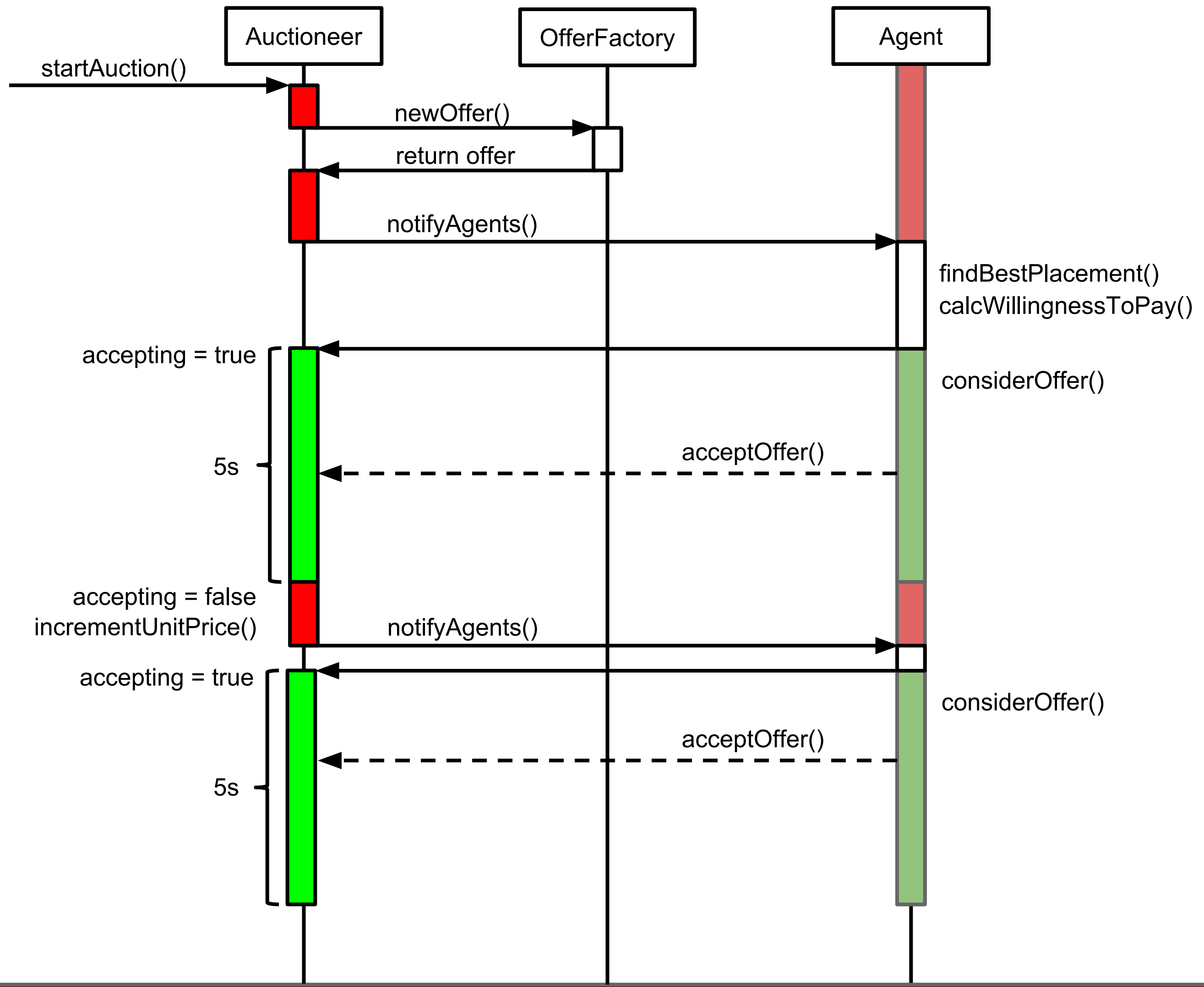
Application Structure

Business Information Systems & Operations Research



Application Structure

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN
Business Information Systems & Operations Research



Implementation Features

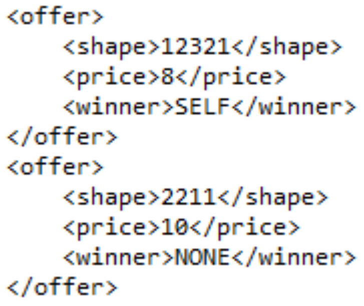
Technische Universität Kaiserslautern
Business Information Systems & Operations Research

File-based Memory (XML)

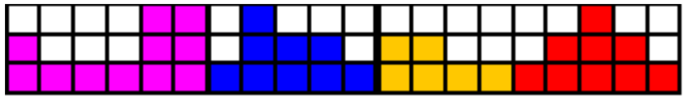
Storing information for price calculation about:

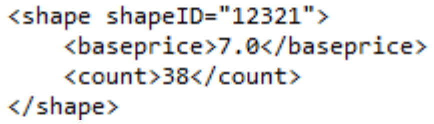
➔ Offer shapes

➔ History of offers & auctions



```
<offer>
    <shape>12321</shape>
    <price>8</price>
    <winner>SELF</winner>
</offer>
<offer>
    <shape>2211</shape>
    <price>10</price>
    <winner>NONE</winner>
</offer>
```





```
<shape shapeID="12321">
    <baseprice>7.0</baseprice>
    <count>38</count>
</shape>
```

Order Placement Algorithm

➔ Keeps even height (from left to right)

➔ Considers blocking of future offers

Implementation Features - Pricing Logic

Pricing Algorithm

1. Get an initial price **based on shape** as **BasePrice**
2. Modify **BasePrice** for the **current offer** (only valid for this offer)
    a. Ranking of shapes
    b. Positioning of shape
    c. Free Space available
    d. Offer history (of the same shape)
    e. Remaining time
3. Modified **BasePrice** as **WillingnessToPay** for **current offer**

➔ Step 1 and step 2 a+d use memory information

➔ **BasePrice** might be adjusted after auction

Demonstration

Improvements

Pricing

→ Optimize weights of price modification

→ Additional factors for modification

- Storing more information as memory

Placement

→ Use more advanced algorithm

Misc

→ Make implementation more generic

→ Replace file-based memory with database

- Use DB for statistical evaluation

Conclusion

Experiences & Knowledge gained

- Modeling a concept for an agent
- Creating a pricing logic
- Refactoring provided code

Difficulties

- Aspects of a (good) decision logic
- Implementation: Idea & Concept → working code
- Implementing auction mechanisms

Finally

Questions?

Source code available at: https://bitbucket.org/lslersch/mas-project