Ontogrid's Negotiation Service integration with GRAAP's WS-Agreement

Shamima Paurobally
University of Westminster
S.Paurobally@westminster.ac.uk
Michael Wooldridge, Valentina Tamma
University of Liverpool

Motivation

- Ontogrid's Negotiation Service has negotiation protocols + strategies
  - Would benefit from a well-defined contract structure
- WS-Agreement has a well-defined template
  - Would benefit from negotiation mechanisms for a process to reach the agreement
- Next Step of post-Ontogrid: Integrate Ontogrid's negotiation service with WS-Agreement?

Objective of this Presentation

- Describe the implementation of our negotiation service
- Propose the next step: integrate with WS-Agreement

One of the requirements of WS-Agreement:

"**Must be composable with various negotiable models:** it must be possible to design negotiation protocols which compose with schemas defined by WS-Agreement"

Aims and Objective for Negotiation in OntoGrid

- Deploy MAS cooperation techniques in (semantic) Grid
- *Not* the aim:
  - "put agents in the Grid"
    - we are *non-intrusive*
- Area of specific attention:
  - *negotiation & agreement*
- Develop semantic grid services that enable software components to coordinate and negotiate to satisfy their overall goals

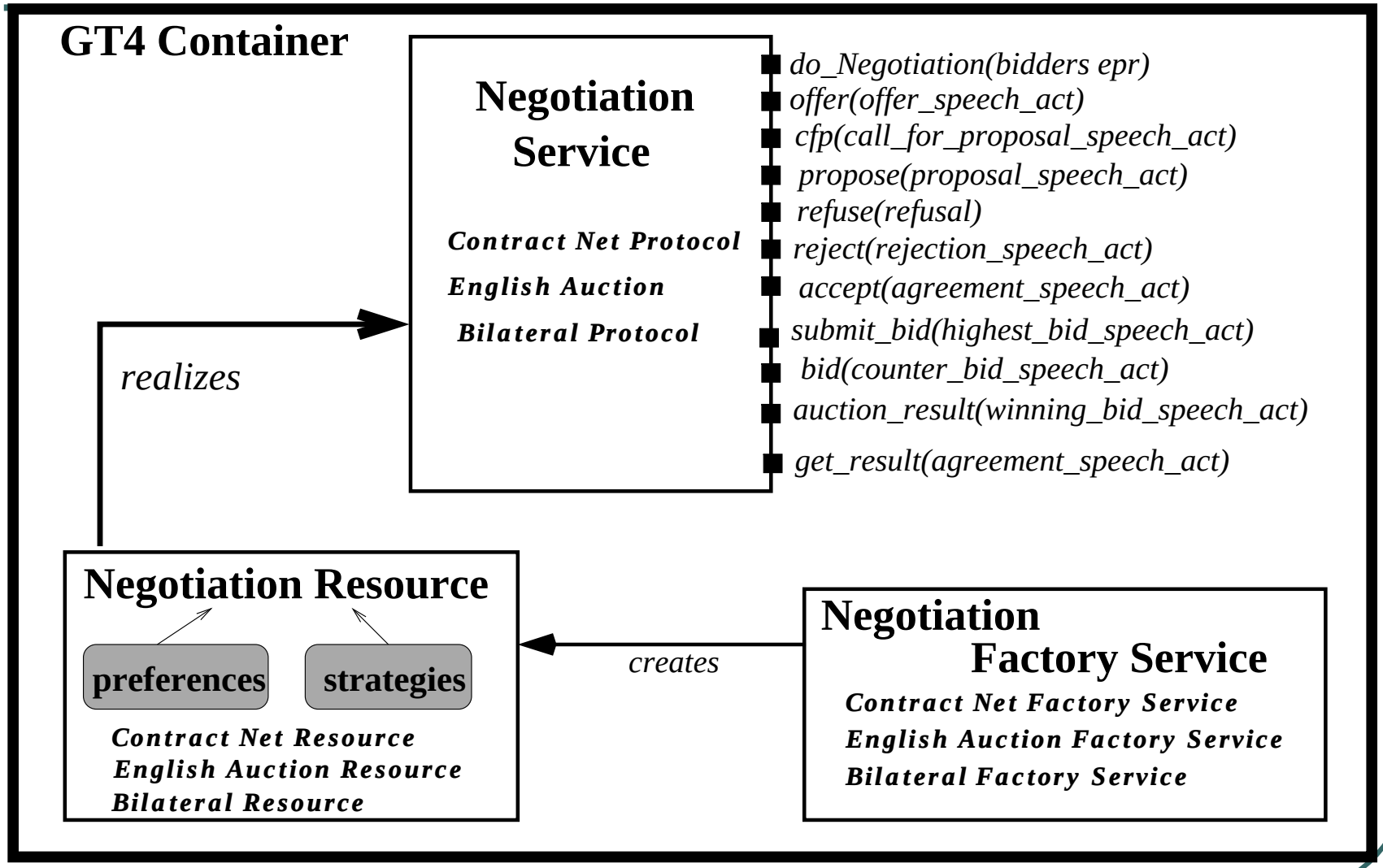
Implementation of WS-Negotiation

- Prototypes implemented for:
  - ✓ Contract Net Protocol for task allocation
  - ✓ Bilateral bargaining protocol
  - ✓ English auction with timeouts for resource allocation
- Deployed in Apache Axis and Tomcat (Version 1)
- Reusable GT4 implementation (Version 2)
- Integrated with OntoGrid architecture
- Deployed in Car Repair Grid

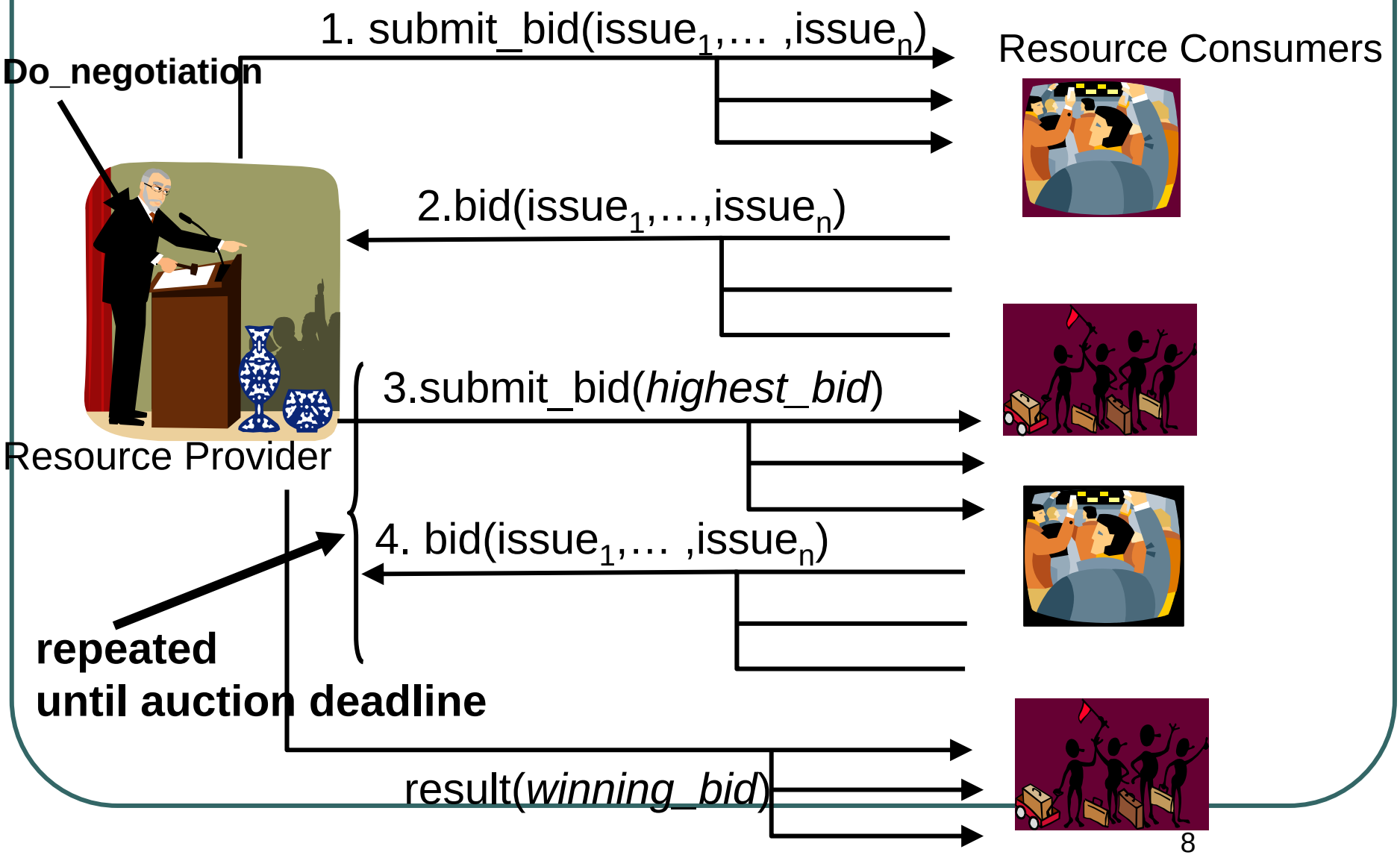
Four Elements to a Negotiation

1. Messages that can be exchanged (public)
   - Port-type of web service e.g. offer, bid, accept, cfp, propose, submit_bid
2. Negotiation protocols (public)
   - Sequence of invoking the methods e.g. provider cfp → consumer propose → provider accept → consumer inform
3. Preferences (private)
   - To decide what makes a good deal e.g. reserve prices
4. Decision strategies (private)
   - To evaluate and generate the content of the messages e.g. time dependent concession

Architecture of the Negotiation Service

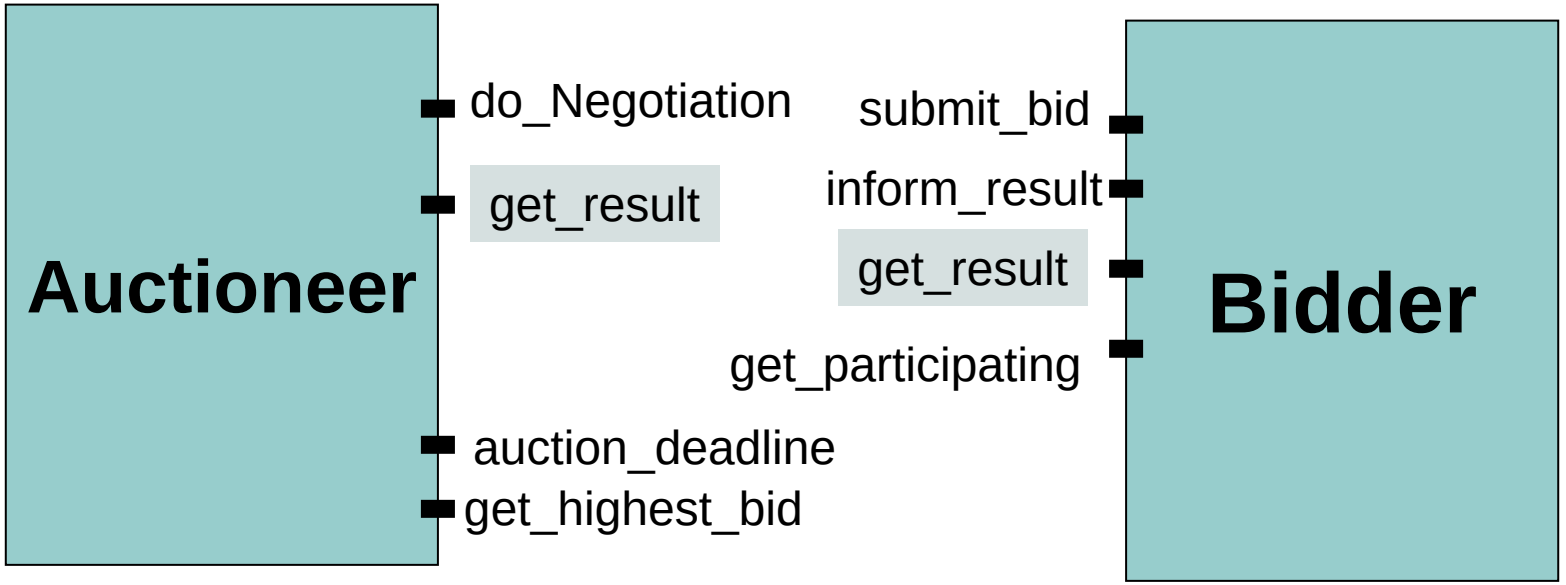


English Auction for Resource Allocation in the Grid



Element 1: Exposed Methods

Methods that can be invoked on a negotiation-capable web service defined in WSDL



DoNegotiation_List

```
<xsd:element name="doNegList">
  <complexType name="doNegListType">
   <sequence>
    <element name="context_job" type="xsd:string"/>
    <element name="AuctionDeadline" type="xsd:int"/>
    <element name="RoundDeadline" type="xsd:int"/>
    <xsd:element ref="tns:bidder_list"/>
    <xsd:element ref="tns:NameofIssuesList"/>
    </sequence>
  </complexType>
  </xsd:element>
```

Example of Name of Issues List: {price, responseTime, statementNumber,…}

Methods Parameters: Speech Act Subject

```
<xsd:element name="Speech_Act_Subject">
  <complexType name="Speech_Act_Sub">
  <sequence>
      <element name="sender" type="wsa:EndpointReferenceType"/>
     <element name="context_job" type="xsd:string"/>
     <xsd:element ref="tns:IssuesList"/>
     <element name="bid_number" type="xsd:int"/>
     <element name="deadline" type="xsd:int"/>
    </sequence>
 </complexType>
</xsd:element>
```

IssuesList is a list of tuple issues {(name, value, isNegotiable),….}

Example (EPR of provider, JobID YU7,
{(price,£20,true), (response,20ms,false)}, bidNo 3, 1000ms)

Bidder and Auctioneer API

do_Negotiation(DoNegList doNegotiation_List)

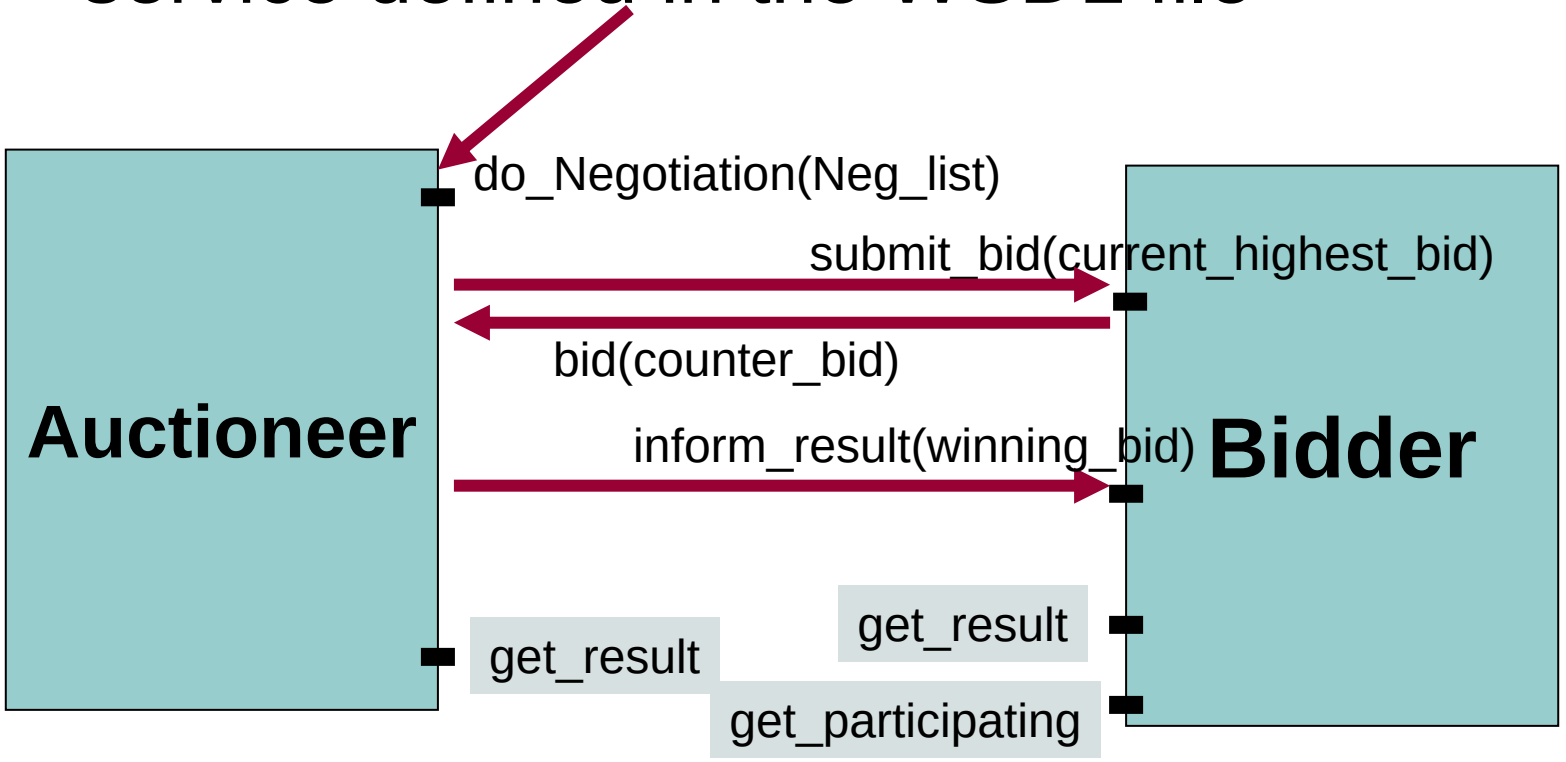
Speech_Act_Subject
submitBid(Speech_Act_Subject highest_bid)

Speech_Act_Subject
informResult(Speech_Act_Subject winning_bid)

ResultNegBean get_result(ContextJob)

Element 2: English Auction Protocol

- Methods that can be invoked on a web service defined in the WSDL file



Business Logic of Auctioneer

do_negotiation(Do_neg_list)

Auctioneer Factory

Initialise auctioneer resource

Delegate auction to auction house

Get preferences of auctioneer from database

WS-DAIONT

Get strategy of auctioneer

Decision Strategies

Auction House

Auction House (Auction Rounds)

```
public class AuctionHouse implements Runnable{
public void run(){
    inform bidders of start of the auction;
    highest_bid = empty starting bid;
        while the auction deadline is not reached {
            for each bidder, call submit bid(highest bid) on each bidder;
                wait for round deadline;
                if any bids have been submitted {
                    highest bid = evaluate best bid(list received bids,
                                                    auctioneerPreferences);}
                else no bids have been received, break;
    } //end while, auction has ended
    overall winning bid = highest bid of last round;
    check that overall winning bid is within reserve preferences;
    inform all bids of the overall winning bid and bidder;
} }
```

Business Logic of Bidder

submit_bid(highest_bid)

Bidder Factory

Initialise bidder resource

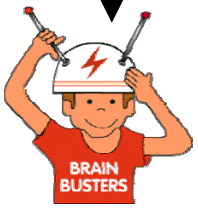
Evaluate highest bid and generate counter_bid

Get preferences of bidder from database

WS-DAIONT

Get strategy of bidder

Decision Strategies



Decision Making Algorithms

Element 3: Preferences Ontology

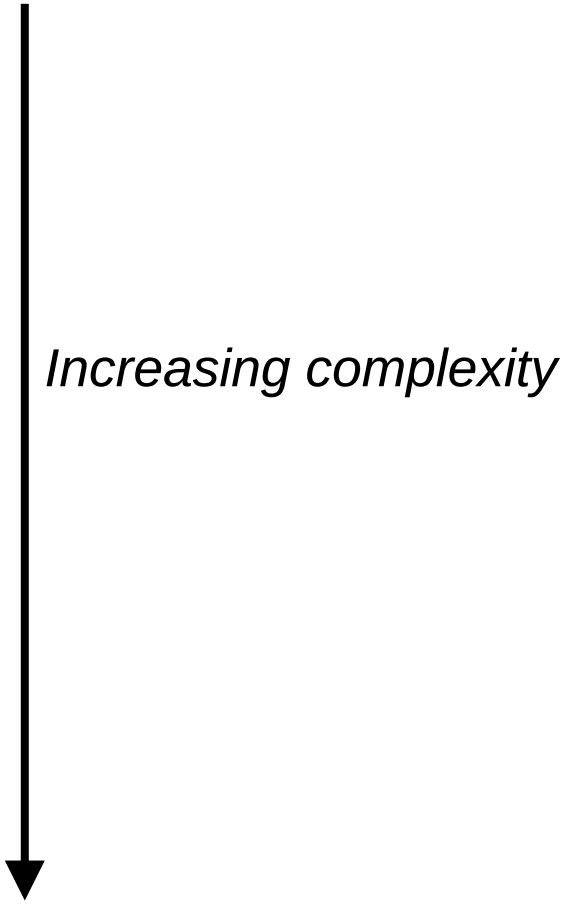
- Preferences capture a user's profile and are stored in distributed databases in Ontogrid's WS-DAIOnt
- Preferences for each issue
  - Issue Name e.g. price
  - Preferred value e.g. A seller has a preferred value of £30 for price
  - Prefers High or Low e.g. A seller prefers high value for price and so will concede in a negotiation
  - Reserve value (maximum or minimum value) e.g. A seller has minimum value for price
  - Is Negotiable e.g. price is negotiable, colour of a car is non negotiable
  - Weight of issue (normalised)
    - If weight of price issue is 0.7 and #statements is 0.3, then price is more important
  - Utility (normalised)
    - e.g. how useful is £30 for price for a seller (could vary with time, resources)

Element 4: Decision Making Strategies

- Auctioneer to evaluate bids and choose highest bid. Calculated from:
  - List of received bids
  - Auctioneer preferences
- Bidder to evaluate current highest bid and generate counter bid. Calculated from:
  - Current highest bid
  - Bidder preferences
  - Auction history
  - Auction and bidder deadlines

Implemented Decision Strategies

- Truth-Telling
- Decrement
- Cost Endowment
- Utility evaluation
- Time dependent
- Utility based generation
- Opponent dependent



Example: Utility Strategy

Evaluation of a bid:

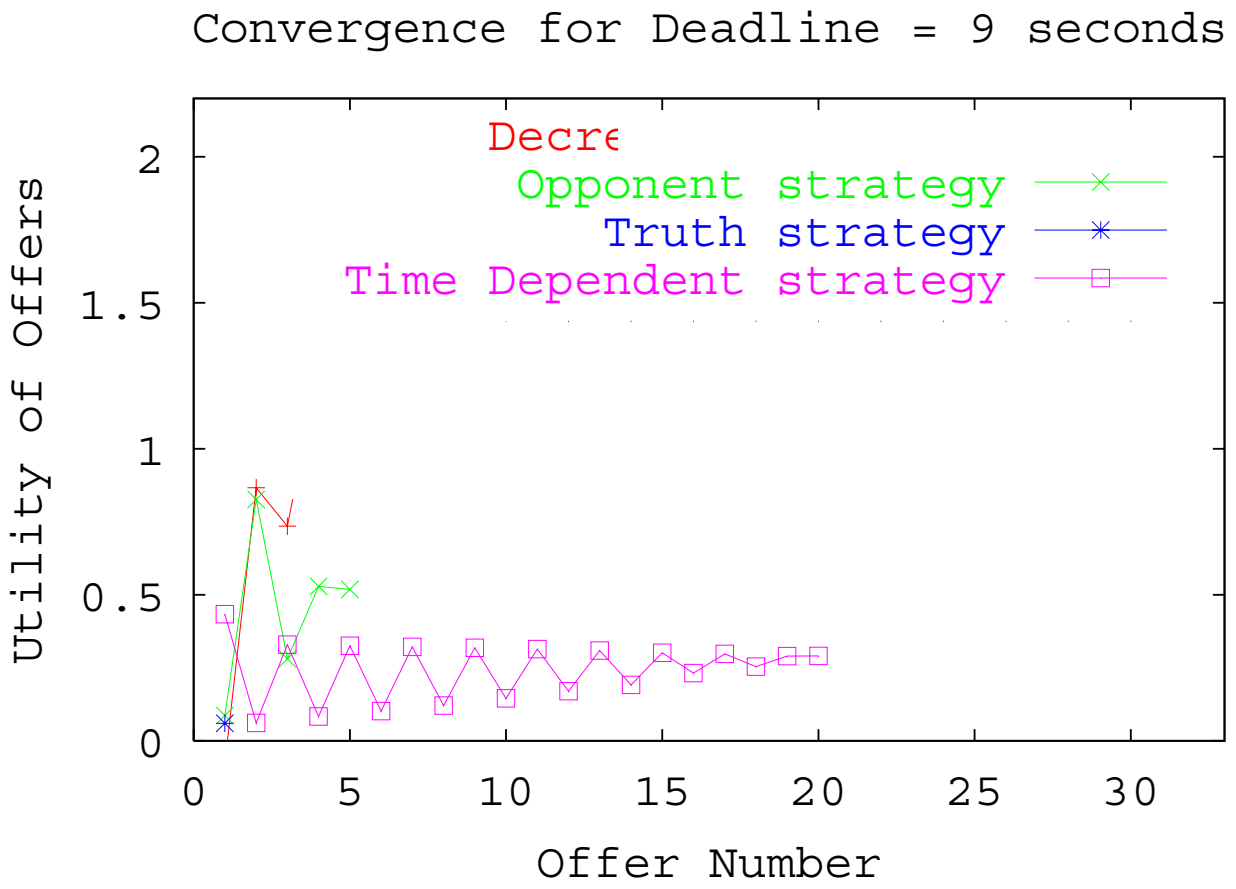
- Evaluation of an issue

$$V_i = \frac{|\, reserve_i - bidValue_i \,|}{reserve_i} \times Utility_i$$

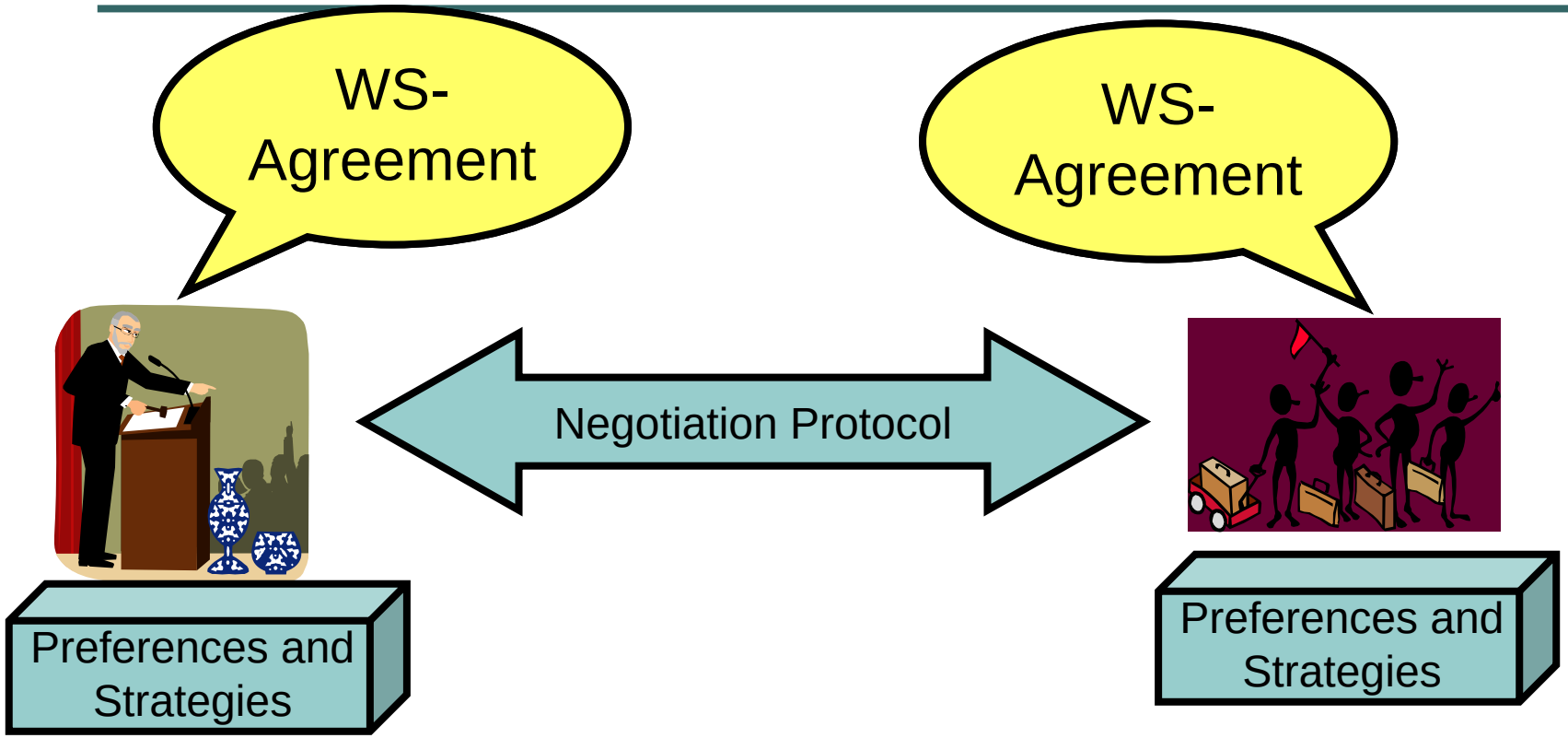
- Score of a proposal

$$V_{proposal} = \sum_{1 \le j \le n} weight_j V_j$$

Example Results – Stability of Market



Next Step: Integrate with WS-Agreement



A bid is of type WS-Agreement instead of type Speech Act Subject

WS-Agreement submitBid(WS-Agreement highest_bid)

WS-Agreement informResult(WS-Agreement winning_bid)

Refactoring the SpeechAct

```
<complexType name=  "Speech_Act_Subject">
<sequence>
    <element name="sender"
 type="wsa:EndpointReferenceType"/>
   <element name=“deadline" type="xsd:int"/>
   <element name="bid_number" type="xsd:int"/>
   <element name="context_job" type="xsd:string"/>

   <xsd:element ref="tns:IssuesList"/>
   </sequence>
</complexType>
```

WS-Agreement

Context

Service Description Term/Name

SDT + service properties

2 Significant Changes

- Changing from IssuesList to SDT and service properties
  - IssuesList is a list of tuple issues {(name, value, isNegotiable),….}
  - How to extract the issues from the resulting WS-Agreement structure for the decision making
- Addition of guarantee terms
  - Currently no guarantee terms in our negotiation
  - But could add them easily if we donot negotiate about them

Possible Future Work

- Translation from speech act subject to WS-Agreement
- 2-part negotiation: negotiation about resources/service and minor negotiation about guarantee terms
  - Our negotiation service has not considered negotiation about guarantee terms. Should there be any negotiation about them?
  - Can they be quantified for decision making?
- Where do the agreement and run-time states fit in?

Thank You