Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2008 Proceedings

European Conference on Information Systems (ECIS)

2008

Semi-Automated Provisioning and Usage of Configurable Web Services

Nikolay Borissov
Information Management and Systems, University of Karlsruhe, Germany, borissov@iism.uni-karlsruhe.de

Benjamin Blau
Information Management and Systems, University of Karlsruhe, Germany, blau@iism.uni-karlsruhe.de

Dirk Neumann
Albert-Ludwigs-Universität Freiburg, dirk.neumann@vwl.uni-freiburg.de

SEMI-AUTOMATED PROVISIONING AND USAGE OF CONFIGURABLE SERVICES

Borissov, Nikolay, University of Karlsruhe, Englerstrasse 14, 76131 Karlsruhe, Germany, borissov@iism.uni-karlsruhe.de

Blau, Benjamin, University of Karlsruhe, Englerstrasse 14, 76131 Karlsruhe, Germany, blau@iism.uni-karlsruhe.de

Neumann, Dirk, University of Freiburg, Platz der Alten Synagoge, 79085 Freiburg, Germany, dirk.neumann@vwl.uni-freiburg.de

Abstract

Although the amount of offered Web services is continuously increasing and the Internet of products shifts more and more towards an Internet of services there is still a big lack of efficient service allocation and price determination. Currently, services are mostly charged as flat fees and pay-per-use prices. Albeit static pricing is the most common pricing scheme, dynamic and decentralized service environment requires highly adaptable mechanism for allocation and price determination. As the state and context of services change rapidly prices and allocations are continuously fluctuating. The contribution of this paper is fourfold. Firstly, we introduce three business scenarios for trading web-based services. Secondly, we analyze design requirements for semi-automated trading of web services. Thirdly, we propose and define a model approach for trading Internet services. Finally, using this model, we specify bid extensions for common resource and service description languages. We evaluate our model by analyzing the introduced concepts and giving examples using the specified bidding language. We ran a simulation with the implemented prototype showing the appliance and flexibility of the model.

Keywords: Semantic Model, Service Provisioning, Strategic Bidding, Pricing.

1 INTRODUCTION

In recent years there has been a rapid change in Web service technologies, their provisioning and ways of consumption. Web services are broadly distributed and used by a vast variety of different consumers with different objectives. On the one hand, there are Web service offerings providing functionality for social and communication purpose like Flickr[1] or del.icio.us[2]. On the other hand Web services are used by companies providing valued-added business functionality in all fields of enterprise activities. Some examples are Xignite[3] or Reuters which offer finance services for a large number of different users. Another example is StrikeIron[4] which provides Web services in the field of CRM, Business Intelligence and data cleansing and processing. Another fast growing field of Web service application is cloud computing. Companies like Amazon (EC2) or Google offer Web service access to clouds of computer systems providing computing power for end-users and businesses.

Although the amount of offered Web services is continuously increasing and the Internet of products shifts more and more towards an Internet of services there is still a lack of efficient service allocation and price determination. Currently services are mostly charged flat fees and pay-per-use prices. Static pricing is the most common used pricing scheme. This fact is controversial because especially in short-living and highly-adaptable environments such as service markets there is a great demand for mechanisms providing dynamic and fast changing service allocation and pricing.

The so-called Internet of services requires models for automated and economically efficient allocation of distributed web services. Moreover, to provide and retrieve services one requires common service ontology – common semantic model of services and their attributes as well as a language to describe the service offers and bids. Autonomous agents trade the providers' services on a web service market and also allocate services for consumers' applications on demand.

The paper is structured in eight sections. In section 2, we introduce three business scenarios for trading web-based services. The first scenario describes a situation, where a provider offers a raw service such as computing power or storage. In the second scenario, an application service is allocated among service requesters. The third scenario illustrates a situation, where a consumer composes a complex service from decentralized providers. In section 3, we analyze design requirements for our service market model approach. Based on the business models and design requirements, section 4 introduces a model approach for semi-automated bidding. In section 5, we introduce schema extensions for a bidding language of common resource and service description languages. In section 6 we present consumer and provider policies for automated bidding. Finally, in section 7 we evaluate our work and section 8 closes with a conclusion showing open challenges and future work.

2 BUSINESS SCENARIOS

In this section we discuss three use-case scenarios for provisioning and usage of distributed (Web) services. Figure 1 illustrates the communication between service providers and consumers common to all three scenarios. A service provider is specifying an *offer* by defining a service description and pricing information. On the other side, a consumer requests a service by describing his preferences in corresponding service and quality parameters with an assigned price and submits it in a *bid*.

The matching process between *offers* and *bids* is provided through a decentralized matching service, called *Service Market* (*SEM*). The objectives of the SEM is *(i)* to reveal the supply and demand of computational services and *(ii)* to efficiently match descriptions of providers' offers and consumers'

[1] http://flickr.com
[2] http://del.icio.us
[3] http://xignite.com
[4] http://strikeiron.com

bids with respect to their technical and economic descriptions. The technical description includes input and output interfaces, service attributes, hardware characteristics and quality parameters. The economic description consists of attributes like provider's reservation price, consumer's willingness to pay, currency and bid increment. The SEM service will support the matching process through common market mechanisms, e.g. Double, English, Dutch auction and their derivates (Friedman 1984, Wurman 1999) as well as provide the capability to attach custom market mechanisms.

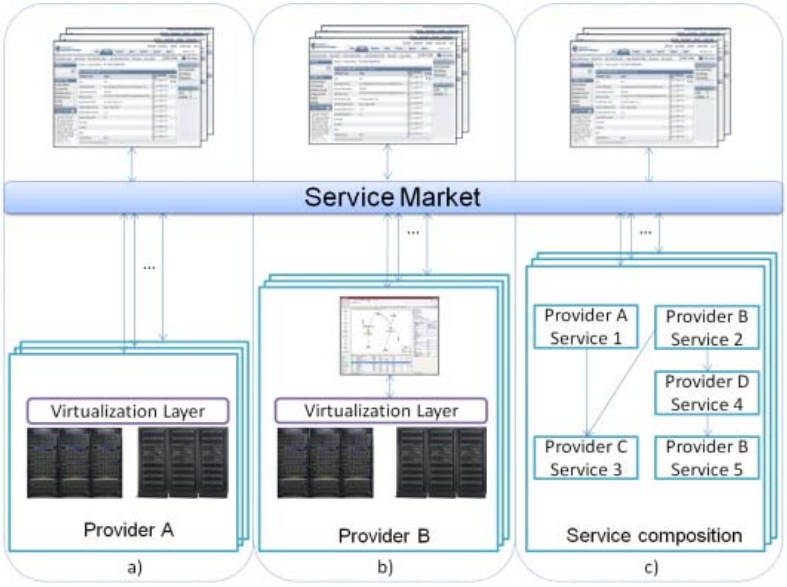

Figure 1. Business use cases for allocation of Internet services.

2.1 Scenario 1 - Raw Service Provisioning

Scenario 1 illustrates the provisioning and usage of pure computational services – *Raw Services* (*RS*) – like CPU, storage and bandwidth. The access to such raw service is enabled through a Web service interface. Prominent examples for such kind of services are Amazon EC2[5] (Elastic Compute Cloud) or Sun's Grid Computing Utility[6]. A service provider specifies the technical description of the offered *raw service* configuration in terms of service attributes for CPU, memory and storage. The offer is comprised of a charge fee for the service usage in terms of valuation (reservation price) and currency.

Analogously, to execute an application (job), a consumer specifies preferences for the requested CPU, memory and storage characteristics as well as her willingness to pay in terms of valuation and currency. Additionally, she specifies input data – executable, data files and required third party libraries as well as output data formats and target place for the output result.

The SEM Web service is responsible for matching of provider's offers and consumer's bids. The matching process is separated into two steps. First, the SEM matches the technical description part of providers' offers and consumers' bid. The second step performs an economic matching of offers and bids resulting in an allocation of services to consumers' requests and price determination for the usage.

2.2 Scenario 2 - Application Service Provisioning

Second business scenario describes a situation where a provider offers an *application* (Web) *service (ApS)* e.g. database, image rendering, video processing, data mining, financial and backup services. Examples in this field are StrikeIron[7] which provides Web services in the field of CRM, Business

[5] http://www.amazon.com/gp/browse.html?node=201590011
[6] http://network.com
[7] http://strikeiron.com

Intelligence and data cleansing and processing or Reuters[8] which offers professional service-based solutions for financial data processing and analysis. In our scenario, the provider publishes a service offer with a description of the service in terms of *service* and *quality attributes*, *input* and *output interfaces*, *data formats* as well as *pricing information*. For the description of application services we postulate the existence of a *core service ontology*, which provides taxonomy of basic services consisting of *core service categories* and *service attributes* enriched with semantic information. To ensure that the offered service meets the promised quality of service, the provider can flexibly allocate raw services (section 2.1) on demand. The provider price for the offered application service is calculated by a function depending on the raw service price and the price for the application. In order to satisfy varying consumer requirements, the application service provider offers her application with different levels of quality of services and prices in conjunction to her business model (pricing policy).

A consumer will describe her preferences for the required application service using the same service ontology. The consumer preference includes the requested application service type (e.g. video rendering service), quality parameters (150f/s etc.) and her willingness to pay for this service.

The matchmaking process of the SEM service is more complex than the first scenario, since it has to match more service and quality attributes of the offers and bids.

2.3 Scenario 3 -Complex Service Provisioning

Scenario 3 addresses the case, where a consumer requests the composition of interrelated, compatible (sections 3 and 4) Web services – *complex services (CS)*. An example is the service-oriented business solution Business-by-design[9] by SAP that enables personalized composition of enterprise services. Services can be offered by one or multiple providers. To enable the composition of Web services, we assume that each service i) is accessible through *standardized input* and *output interfaces*, *data formats* and ii) there is a standardized *core service ontology (CSO)* for *service* and *quality attributes*.

Using the CSO, the consumer will specify a set of the requested service types with desired preferences, their interdependencies in terms of input and output interfaces and data formatted as well as the valuation for the whole requested service composition. The SEM will validate the specified service types and will return an allocation only if the services are compatible (section 4).

Based on the CSO, each service provider will offer a well-defined description of her service on the SM. The provider is incentivized to assure service compatibility among the offered services since she can only increase the usage and hence the profit of her service.

In this case, the matchmaking mechanism of the SEM has to allocate the requested service bundle or provide alternative configurations and price offers to the consumer.

3 REQUIREMENTS

Based on the introduced scenarios we derive a set of requirements for semi-automated Web service provisioning and usage. The derivation is based on related works in the fields of web services composition and matching (De Roure et al. 2005, Sycara et al. 2003, Roman 2005), policy-based matching (Berardi et al. 2006, Lamparter and Ankolekar 2007, Milanovic et al. 2003, Stollberg 2005) and business process description and discovery (Benatallah et al. 2005, Sirin et al. 2002).

- R1 - *Service Configurations*: This requirement addresses the ability of a service provider and consumer to specify service configurations at different levels of service quality and price. For example providers of video processing services need to be able to describe (in sense of input and output interfaces) different service configurations to provide desired quality level for a requested price. Moreover, service consumer can require a service with less quality for a lesser price.

[8] http://about.reuters.com/productinfo/financial/
[9] http://www.sap.com/solutions/sme/businessbydesign/

- R2 - *Context-dependent Preferences*: Service consumers have different preferences about the services they are willing to use, location, qualities, etc. a consumers request and choose a service, taking their own preferences into account, e.g. on service characteristics such as availability, response time, price etc. Therefore, we need a way to describe requests and preferences for particular service configurations declaratively with respect to their attributes.
- R3 - *Business Policies Appliance*: The requirement refers to the ability of consumers and providers to describe preferences, utilities and prices as a function of service attribute values and to use declarative rules to model the context-sensitive nature of preferences.
- R4 - *Semantic Service Descriptions*: Semantic description languages such as OWL-S and WSMO (Dickinson and Wooldridge 2005; Rao et al. 2004) enable the explicit description of a service and its attributes. Moreover, semantic technologies enable logical reasoning to bridge different levels of abstraction that occur when specifying offers and bids.
- R5 - *Service composability and compatibility*: Semantic (Web) service description technologies (Haller et al. 2005) enable modularization and composition of distributed services. However, service composition requires selecting services that are compatible in terms of well defined and standardized interfaces as well as input and output data formats (Stollberg 2005).
- R6 - *Efficient service matching*: Requested service configurations should be efficiently matched in terms of interfaces, input and output data formats (Constantinescu et al. 2004; Pahl 2007) with offered services in a specified service repository.

Following the requirements, section 4 explicitly defines a model for service offering and requesting.

4 ABSTRACT MODEL

In order to perform the task of service selection, one requires *(i)* a facility for communicating service offers and bids, *(ii)* a policy for matching and ranking offers to bids (Lamparter et al. 2007). The proposed model illustrates a core approach for description of service offers and bids over service configurations. For the reader's convenience we use UML class diagrams to define OWL concepts and their properties (Brockmans 2004). The core concepts of the model are *Configuration*, *Bid* and *Policy*.

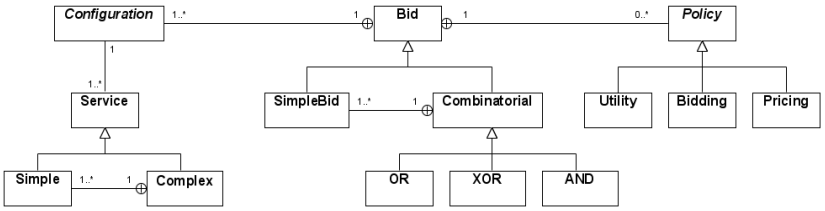

Figure 2. Abstract Model for service offering and requesting.

The concept *Bid* represents both - a service offer and a request. Each bid is associated with a service *configuration*. The semi-automated bid generation process is supported by pre-configured *policies*.

Definition 1. *Configuration:* The concept *configuration* enables including single or collective service description documents into the overall bid document. Moreover the bidder can specify preferences over the requested services using AND, OR and XOR bid constructs.

Definition 2. *Service*: Let $S = (s_1; \dots ; s_{|S|})$ be the set of all services. A service $s_i \in S$ is described by the tuple $s_i = (I_i; O_i; C_i)$ where I_i represents the set of input elements that are required by the service s_i and O_i the set of output elements that are returned by the service s_i, respectively. Furthermore, a service is characterized by a set of feasible configurations C_i with $C_i \subseteq C$, where C is the Cartesian product of the discrete service attributes $A_1, \dots, A_n$, i.e. $C = \prod_{r\in[1,\dots,n]} Ar$. In this context, each $c \in C$ is a vector $c = (a_{1k_1}, \dots, a_{jk_j}, \dots, a_{nk_n})$ where a_{jk_j} represents the k-th value of attribute j.

Bidding languages are well-established means within the economic literature (Cerquides 2007, Nisan 2006,Wurman 1999). Based on these works, we explicitly define a model approach for bid generation.

Definition 3. *Bid*: Assume a set of providers P and consumers U as well as a set of bids B . A service bid is characterized by a vector $b_p = (s_p, G_p)$, where $s_p \in S$ represents a service description and $G_p : C_p \to \Re$ the pricing function that assigns a price to each configuration $c \in C_p$ of the service s_p. We assume that the pricing function is described by an additive function, where g_i represents the pricing function of provider p or consumer u for attribute j. Weight function w_j^g can be used to adjust the influence of the different attributes on the price.

$G_p(c) = G_p\left(a_{1k_1}, \dots, a_{jk_j}, \dots, a_{nk_n}\right) = \sum_{j=0}^{n} w_j^g g_j\,(a_{jk_j})$ with $\sum_{j=0}^{n} w_j^g = 1$

Definition 4. *Simple bid:* A simple bid consists of a (c_i, π_i) pair, where c_i is the offered or requested service configuration and π_i is the provider's reservation price or consumer's willingness to pay.

A consumer can allocate computational services from more than one provider, e.g. she can obtain a combination of services – CPU service from Sun and store the data using Amazon's S3.

Definition 5. *Combinatorial bid*: A combinatorial bid is a bid of the form $(\{\Sigma\ \{op\ \{\Sigma\ \{op\ \Sigma\}^*\}^*\}^+)$, where $\Sigma = C_i\ op\ C_j\ ,\ C_{i,j} \in C$ is a service configuration and $op \in \{OR, AND, XOR\}$. In particular OR bids are not capable of representing single item valuations even on two items, moreover it represents the aggregated willingness to pay for a certain subset. Using AND-bids, the bidder is willing to obtain all of the specified configurations at once, because CPU without storage is not worth for his application. The XOR construct implies that the bidder is willing to obtain at most one of these service configurations for the specified price.

Definition 6: *Policy:* The policy component is a logic part of the bid generation process. The *pricing policy* represents the price building function for a given configuration, the *bidding policy* is a decision function, which determines the bidding strategy. With *utility policies*, providers and consumers can specify criteria, which have to be maximized by the bidding process.

Definition 7. *Service Compatibility*: Let s_r be a service of provider r with the corresponding input I_r and output interfaces O_r. Analogously, s_p is a service of provider p with the inputs I_p and outputs O_p. Then, the service s_r is compatible to s_p iff $I_p \in I_r$ and $O_r \in O_p$. A service compatibility is indicated by the notation $s_r =_c s_p$. The set $C^r \in C$ contains all services that are compatible with a given service description s_r, i.e. $C^r = \{s_p \in C | s_r =_c s_p\}$.

Definition 8. *Technical Matching*: Let b_r be a consumer bid for service s_r with the corresponding service description attributes A_r. Respectively, let b_p be a service offer containing the service s_p with the attributes A_p. Then, b_r technically matches b_p iff $A_r \in A_p$ and $s_r =_c s_p$. A technical match is indicated by notation $b_r =_t b_p$. The set $B_t^r \in B$ contains the service bids that technically match a given consumer bid b_r, i.e. $B_t^r = \{b_p \in B | b_r =_t b_p\}$.

Definition 9. *Economic Matching*: Let b_r be a consumer bid for service s_r with the corresponding price π_r. Analogously, b_p is a service offer containing the service s_p with price π_p. Then, bid b_r economically matches b_p iff $b_r =_t b_p$ and $\pi_r > \pi_p$. An economic match is indicated by notation $b_r =_e b_p$. The set $B_e^r \in B$ contains the service bids that economically match a given consumer bid b_r, i.e. $B_e^r = \{b_p \in B | b_r =_e b_p\}$.

5 SCHEMA EXTENSION FOR BIDS

In this section we propose a schema extension for *bids* which is applied but not limited to de-facto XML-based description languages for job description and submission like JSDL (Anjomshoaa et al. 2005), JDL-GLUE (Laure et al. 2006) and WS-Agreement (Andrieux et al. 2007). WS-Agreement is a web services protocol for establishing agreement between two parties and therefore more suitable for negotiations than for auctions. WS-Agreement aims for a more generic level, thus service description in JSDL and JDL-GLUE can be included as part of it.

We consider the following language extensions for a bid specification:

- *Type:* bid or offer
- *Participant:* a unique identification of the market participant
- *Price:* Represents the provider's, respectively consumer's price; In case of a deal, it represents the final price a consumer is paying to the provider; measurement unit is cent
- *Duration:* Represents the requested computational time measured in milliseconds
- *TimeToComplete:* represents information about a tentative or agreed completion time which can be reported from the provider. Optional component depending on the market mechanism
- *Currency:* Each valuation is associated with a currency
- *TimeValid:* The time in milliseconds a bid is valid. This extension is optional and depends on the used market mechanism (e.g. a continuous double auction)
- *Configuration:* This element provides the ability to bid for single or combination of service configurations specified with XOR, OR and AND bid constructs.

We propose following schema extensions for JSDL, JDL-GLUE and WS-Agreement:

```
< xsd:complexType name = "Bid_Type" >
    < xsd:sequence >< xsd:element ref = "price" minOccurs = "1" maxOccurs = "1"/>
                    < xsd:element ref = "duration" minOccurs ="1" maxOccurs = "1"/>
                    < xsd:element ref = "timeToComplete" minOccurs ="1" maxOccurs = "1"/>
                    < xsd:element ref = "timeValid" minOccurs = "0" maxOccurs = "1"/>
                    < xsd:element ref = "type" minOccurs = "1" maxOccurs = "1"/>
                    < xsd:element ref = "participant" minOccurs = "1" maxOccurs = "1"/>
                    < xsd:element ref = "Configuration" minOccurs = "1" maxOccurs = "1"/>
    </xsd:sequence ></xsd:complexType >
< xsd:complexType name = "Configuration_Type" >
    < xsd:sequence >< xsd:element ref = "XOR"/>
                    < xsd:element ref = "AND"/>
                    < xsd:element ref = "OR"/>
                    < xsd:element ref = "service"/>
    </xsd:sequence ></xsd:complexType >
< xsd:complexType name = "XOR_Type" >
    < xsd:sequence > < xsd:element ref = "XOR"/>
                    < xsd:element ref = "AND"/>
                    < xsd:element ref = "OR"/>
                    < xsd:element ref = "service"/></xsd:sequence ></xsd:complexType >
< xsd:complexType name = "AND_Type" >
    < xsd:sequence >< xsd:element ref = "XOR"/>
                    < xsd:element ref = "AND"/>
                    < xsd:element ref = "OR"/>
                   < xsd:element ref = "service"/></xsd:sequence ></xsd:complexType >
< xsd:complexType name = "OR_Type" >
     < xsd:sequence >< xsd:element ref = "XOR"/>
                    < xsd:element ref = "AND"/>
                    < xsd:element ref = "OR"/>
                          < xsd:element ref = "service"/></xsd:sequence ></xsd:complexType >
< xsd:complexType name = "Service_Type" >
     < xsd:sequence >< xsd:attribute name = "serviceRef" type = "xsd:IDREF"/></xsd:sequence >
</xsd:complexType >
< xsd:element name = "Bid" type ="Bid_Type"/>
< xsd:element name = "configuration" type ="Configuration_Type"/>
< xsd:element name = "XOR" type ="XOR_Type"/>
< xsd:element name = "AND" type ="AND_Type"/>
< xsd:element name = "OR" type ="OR_Type"/>
< xsd:element name = "service" type ="Service_Type"/>
< xsd:element name = "price" type ="xsd:double"/>
< xsd:element name = "duration" type ="Service_Type"/>
< xsd:element name = "timeToComplete" type ="xsd:double"/>
< xsd:element name = "timeValid" type ="xsd:long"/>
< xsd:element name = "type" type ="xsd:string"/>
< xsd:element name = "participant" type ="xsd:string"/>
```

6 POLICIES FOR AUTOMATED BIDDING

Within this section, we present policies for the automated bid generation process. With pricing policies providers and consumers specify how the price for a given service configuration and quality is calculated. Utility policies specify the utility maximization functions. The bidding policy determines the bidding strategy i.e. the rules how, what and until when to bid in respect to the market mechanism.

6.1 Pricing and Utility Policies

In the following pricing function *machinePrice*, a raw service (RS) or application service (ApS) price is calculated based on the service utilization – a common metric for pricing of computational resources (Mendelson et al. 1985), the service's reservation price and an increment. The machine price is dynamically adjusted based on pre-configured increments I_j and utilization bounds U_j (30-50%, 50-70% etc.). Following code snippet represents the pricing policy in JESS (Friedman-Hill et al. 2003):

```
(deffunction machinePrice (?reservationPrice ?utilization ?U1 ?U2 ?U3 ?I1 ?I2 ?I3)
  (if (and (>= ?utilization 0)(< ?utilization ?U1))          then (return (+ ?reservationPrice 0))
    else (if (and (>= ?utilization ?U1)(< ?utilization ?U2))  then (return (+ ?reservationPrice ?I1))
    else (if (and (>= ?utilization ?U2) (< ?utilization ?U3)) then (return (+ ?reservationPrice ?I2))
    else (return (+ ?reservationPrice ?I3))))))
```

The consumer's pricing policy *buyerPrice* returns a bid price based on the consumer's application *valuation* and the actual *releaseDate*. We assume that RS or ApS are mostly utilized between 7am and 8pm, so the consumer's valuation may depend on the urgency of her application. This pricing policy defines time-periods T_j, $T \in [0; 24)$ during which the price is (negative) incremented by I_j:

```
(deffunction buyerPrice (?valuation ?releaseDate ?T1 ?T2 ?T3 ?T4 ?I1 ?I2 ?I3)
  (if (and (>= ?releaseDate 0)(< ?releaseDate ?T1))          then (return (+ ?valuation 0))
    else (if (and (>= ?releaseDate ?T1) (< ?releaseDate ?T2)) then (return (+ ?valuation ?I1))
    else (if (and (>= ?releaseDate ?T2) (< ?releaseDate ?T3)) then (return (+ ?valuation ?I2))
    else ((+ ?valuation ?I3))))))
```

We assume that the objective of a service provider is to maximize the user's satisfaction as well as her profit. A simple utility function which provides fair price-based scheduling based on the consumer's valuation v for a requested processing time p is presented by Heydenreich et al.: $U_p = max \sum_j \frac{v_j}{p_j}$

Using this utility policy, the allocated jobs are prioritized in the provider's machine queue according to the ratio of consumer's valuation and requested processing time. A consumer is willing to execute her application (job) using a provider service. Ideally the job is allocated to a service, so e.g. the completion time (or some other objective) is minimized. Self-interested consumers, however, will be interested in minimizing their own completion time C and payments π based on their application's valuation *v*. The following consumer's policy (Heydenreich et al. 2006) implements this objective: $U_j = max \sum_m -v_j C_{j,m} - \pi_{j,m}$

6.2 Bidding Policies

In the literature (Phelps 2007, Tenorio 1999) bidding strategies are commonly classified in:

- *Truth-Telling Strategy*: The truth-telling strategy places a bid equal to the provider's or consumer's valuation. Although it is a simple strategy, truth-telling is of fundamental importance, since in incentive-compatible mechanisms, this strategy guarantees to obtain the optimal pay-off for consumers no matter what strategies are adopted by the others. In budget-balanced double-auction mechanisms, on the other hand, this strategy is not dominant.
- *Equilibrium-Price Strategy*: Describes the case when consumers or providers hypothetically knowing the true equilibrium price of a service, so they can coordinate high efficiency outcomes in a wide variety of mechanisms regardless of incentive-compatibility properties.

- *Reinforcement-learning Strategies*: These strategies rely on feedback from interacting with the mechanism and can be used in any auction-based mechanism with scarce or no access of public market data. A prominent example is the Q-learning algorithm. To define a bidding strategy, the algorithm requires a definition of states, the admissible actions in each state, the returned reward in each state as well as the policy for changing the state and selecting the best action. A commonly used greedy policy is to select the action that yields the highest Q-value for a given state.

7 EVALUATION

The section starts with analysis of the proposed model for automated service provisioning and usage and gives an example of the specified bidding language. Finally, it presents simulation results of the implemented prototype.

7.1 Technical analysis

In section 4 we introduced a core model for trading Internet services. Using policies, a provider or consumer is able to configure price, utility functions and bidding strategies for the automated service provisioning and usage (R3). For the concept *configuration* we assume the usage of standardized service ontology and require the *id* of the described service as a part of the bid document. Using the concept *simple bid* a consumer or provider is able to specify a price for a raw service (scenario 1) or application service (scenario 2) configuration, whereas the usage of logical operators XOR, AND or OR enables the specification of more complex service configurations (scenario 3).

In the following example we demonstrate the usage of the bidding language proposed in section 5 in respond to requirements R1and R2. Suppose a consumer is requesting a complex service for the duration of 2 hours, which can be covered either by configurations s_1 or $\{s_2$ AND $s_3\}$, each of them achieving a similar outcome. The maximum consumer's price for one of them amounts 1.8 EUR:

$< ext{:}Bid >< ext{:}price > 1.80 </ext{:}price >$
$< ext{:}duration > 7200000 </ext{:}duration >$
$< ext{:}timeValid > 1000000 </ext{:}timeValid >$
$< ext{:}type > bid </ext{:}type >$
$< Configuration >< XOR >< Service\ serviceRef = "s1" /><\ AND >< Service\ serviceRef = "s2"/>$
$< Service\ serviceRef = "\ s3" /></AND ></XOR ></Configuration >$
$</ext{:}Bid >$

This proposed bidding language extension (section 5) enables the specification of single price for single service configuration, but also for combination of configurations. Based on specified price the market mechanism should find a suitable service allocation and respectively calculate the payments. The bid and policy components are motivated by the fact that computational, data mining and financial services are often charged for a fee and the increasing demand and supply requires an automated approach for providing and matching of Web services. This is especially essential for companies providing large-scale computational services, but also "large-scale" applications e.g. video-processing, computer aided design and simulations. Using the model and the bidding language, an autonomous agent can be configured with policies and service configurations in order to trade their services automatically on the service market. On the other side such agents can also be used in a similar way by customers who use applications regularly. Application providers using raw services on demand or reusing existing services in order to minimize development costs and time can also benefit from using autonomous agents. This model approach applied to raw services is evaluated in the following section.

7.2 Simulation

The implemented prototype (Figure 3) consists of two types of implemented agents – Job (consumer) agent and machine (provider) agent. Each machine agents uses the same pre-configured pricing policy and each job-agent the same utility policy (section 6), which are loaded into the rule-engine JESS. The agents are adopting the truth-telling bidding strategy for reporting the job requirements and payments.

Figure 3. A prototype implementation of the specified model.

Each job and machine agent submits a bid in form of extended JSDL file (section 5) for a raw service (scenario 1). The service market (SEM) component implements two allocation mechanisms – FIFO as variant of a technical scheduler and a decentralized online scheduling mechanism (DLGM), presented in Heydenreich et al. 2006. The DLGM mechanism aims to minimize the total weighted completion time $\sum w_j C_j$ across all jobs, hence maximizing their utility. Based on the assumptions of DLGM, we consider that the provided machines are homogeneous and their CPU, storage and OS technical description always match the consumer requirements. DLGM comprises following three steps:

- *Step 1 – Job submission*: on a release date r_j of job j a job-agent submits a bid to the SEM requesting a computational service for the duration d_j and waiting costs c_j. The waiting costs of a job express the costs for waiting one additional time unit in the machine's waiting queue. The SEM in the case of DLGM forwards the request to all registered machines $m \in M$.
- *Step 2 – Real-time planning*: Based on the received information, the machines perform real-time planning based on a local scheduling policy – if job *j* has a higher priority value than $k \Leftrightarrow \frac{c_j}{d_j} \geq \frac{c_k}{d_k}$, then *j* is scheduled before job *k* in the waiting queue. Depending on the current local waiting queue, the machine *i* reports a tentative (ex-ante reported) completion time $\tilde{C}$ and payment $\tilde{\pi}$ to the agent of job *j* (Heydenreich et al. 2006). The payment $\tilde{\pi}$ contains the aggregated compensations to all job-agents whose jobs are currently waiting at machine *i* and are delayed due to allocation of *j*.
- Step 3 – *Machine selection*: Upon receiving information about its tentative completion time and required payments, the job-agent makes a binding decision to queue at certain machine *i*, and pays $\tilde{\pi}$ to the delayed jobs.

In order to evaluate the two mechanisms within the prototype, we ran six settings (Table 1), whereas each setting comprised 5 machines agents. Each setting was run in 25 rounds with the duration of 1000 time units. To increase the competition in the market, across the course of these settings, we successively increased the Poisson-based arrival rate of jobs from $\lambda = 0.2$ to $\lambda = 1.0$ as listed in Table 4, which also specifies the random choices of the durations and weights.

Setting	S1	S2	S3	S4	S5	S6
Arrival time $\boldsymbol{\lambda}$ of the Poisson-based release dates $\boldsymbol{r_i}$	.2	.5	.6	.7	.8	1
Actual number of jobs	208	440	559	686	795	1050
Duration $\boldsymbol{d_i}$	Normal distribution with mean 9 and variance 3					
Weighting costs $\boldsymbol{c_i}$	Normal distribution with mean 9 and variance 3					

Table 1. Simulation settings

The results of the simulations are summarized in Table 2. The utilization being defined as the number of jobs where the mechanism is busy divided by the total number of time slots represents a good metric for expressing the level of competition. As all two mechanisms do not impose price bounds like job budgets, utilization is identical for all three mechanisms. For an arrival rate of $\lambda = 0.2$ with a job sequence of 208 jobs, the machines are only used 38 % of the time. When the size of the job sequence increases to 559 jobs, the utilization rate grows up to 77 %, and the machines are fully utilized for settings with more than 700 jobs. When the competition is low, technical schedulers achieve as good welfare (i.e. total weighted completion time) as DLGM does. However, with increasing utilization, technical schedulers are starting to perform worse than market mechanisms. When competition is extremely high, FIFO generates about 13 % higher overall waiting costs than DLGM.

Metric	Mechanism\Setting	S1	S2	S3	S4	S5	S6
Utilization	All mechanisms	38%	77%	91%	100%	100%	100%
Aggregated waiting cost	FIFO	-916K	-1,993K	-2,676K	-3,633K	-5,046K	-8,784K
	DLGM	-916K	-2,001K	-2,650K	-3,340K	-4,370K	-7,042K
Positive Compensations	FIFO	0%	0%	0%	0%	0%	0%
	DLGM	1%	7%	16%	18%	23%	34%
Lateness [Time periods]	FIFO	0	0	0	0	0	0
	DLGM	.002	.073	1.13	8.72	20.73	58.99

Table 2. Simulation results – FIFO vs. DLGM

This result is rather straightforward, as technical schedulers do not account for the relative urgency of jobs, specified indirectly with the application valuation. The lateness metric measures the time difference between ex-post and tentative completion time.

8 CONCLUSIONS AND OUTLOOK

In this paper we introduce three business scenarios for trading Web services, section 2. Based on these, section 3, we derive design requirements for semi-automated trading. Section 4 present and explicitly defines a core model for trading Internet services. Using the model, in section 5 we specify bid extensions for common resource and service description languages as well as present policies for automated bidding, section 6. We evaluate our model, first, by analyzing the introduced concepts and giving an example for a bid specification using the specified extensions. Secondly, we show simulation results from the implemented prototype showing the appliance and flexibility of the model.

Since there is no known ontology, which explicitly defines and categorize service and quality attributes like required in scenarios two and three (R5), the next steps will include the design and development of such simple core service ontology. For the definition of the core concepts and attributes we will employ web service repositories like *seekda* (www.seekda.com) and *QWS* (www.uoguelph.ca/~qmahmoud/qws). Moreover, to evaluate the efficiency and applicability of an economic and technical matching of Web services (R5, R6) we are elaborating algorithms validating the composability and compatibility of service descriptions as well as suitable matching algorithms, in form of market mechanism, for allocating bids and offers of application and complex services.

References

Andreozzi, S.; Ehm, F. and Field, B. (2007), GLUE Specification v. 2.0, GLUE-Working group.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007), Web Services Agreement Specification (WS-Agreement).

Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., and Savva, A. (2005) Job Submission Description Language (JSDL) Specification, p. 15.

Benatallah, B., Hacid, M.S., Leger, A., Rey, C., and Toumani, F. (2005), On automating Web services discovery, The VLDB Journal The International Journal on Very Large Data Bases, pp 84-96.

Berardi, D., Calvanese, D., De Giacomo, G., and Mecella, M. (2006), Automatic Web Service Composition: Service-tailored vs. Client-tailored Approaches, Proc. AISC).

Bichler, M. and Kalagnanam, J.(2005), Bidding Languages and Winner Determination in Multi-Attribute Auctions, European Journal of Operational Research, pages 380-394

Brockmans, S., Volz, R., Eberhart, A. and P. Löffler (2004), Visual modeling of OWL DL ontologies using UML. In S. M. et al., editor, Proc. of the 3rd International Semantic Web Conference, pages 198–213, Hiroshima, Japan, Springer LNCS.

Cerquides, J., Endriss, U., Giovannucci, A., and Rodriguez-Aguilar, J.A. (2007), Bidding Languages and Winner Determination for Mixed Multi-unit Combinatorial Auctions, Proc. of the Twentieth International Joint Conference on Artificial intelligence (IJCAI 2007), Hyderabad. India, January

Constantinescu, I., Faltings, B., and Binder, W. (2004), Large Scale, Type-Compatible Service Composition, Proceedings of the IEEE International Conference on Web Services (ICWS'04).
De Roure, D., Jennings, N., and Shadbolt, N. (2005), The semantic grid: past, present, and future Proceedings of the IEEE (93:3), pp 669-681.
Dickinson, I., and Wooldridge, M. (2005), Agents are not (just) web services: considering BDI agents and web services.
Engel, Y., Wellman, M.P., and Lochner, K.M. (2006), Bid expressiveness and clearing algorithms in multiattribute double auctions, 7th ACM conference on Electronic commerce, pp 110-119.
Friedman-Hill, E., and others (2003), Jess, the Rule Engine for the Java Platform, Distributed Computing Systems.
Friedman, D. (1984), On the Efficiency of Experimental Double Auction Markets, American Economic Review (Vol. 24, No. 1), pp 60-72.
Fu, Y., Chase, J., Chun, B., Schwab, S., and Vahdat, A. (2003), SHARP: an architecture for secure resource peering, ACM SIGOPS Operating Systems Review (37:5), pp 133-148.
Haller, A., Cimpian, E., Mocan, A., Oren, E. and Bussler, C. (2005), WSMX-a semantic service-oriented architecture, Web Services, ICWS 2005. Proceedings. 2005 IEEE International Conference on pp 321-328.
Heydenreich, B., Müller, R., and Uetz, M. (2006), Decentralization and Mechanism Design for Online Machine Scheduling, Maastricht research school of Economics of TEchnology and ORganizations.
Lamparter, S., and Ankolekar, A. (2007), Automated Selection of Configurable Web Services, 8. Int. Tagung Wirtschaftsinformatik).
Lara, R., Roman, D., Polleres, A., and Fensel, D. (2004), A Conceptual Comparison of WSMO and OWL-S, Proceedings of the European Conference on Web Services (ECOWS 2004)).
Laure, E., Fisher, S., Frohner, A., Grandi, C., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., and others (2006), Programming the Grid with gLite, Computational Methods in Science and Technology (12:1), pp 33-45.
MacKie-Mason, J.K., and Wellman, M.P. (2006), Automated Markets and Trading Agents, in: Handbook of Computational Economics, vol. 2: Agent-Based Computational Economics, L. Tesfatsion and K.L. Judd (eds.), North-Holland.
Milanovic, N., Stantchev, V., Richling, J., and Malek, M. (2003), Towards Adaptive and Composable Services, Proceedings of the IPSI2003).
Nisan, N. (2006), Bidding languages, Combinatorial auctions, Cambridge, MIT.
Pahl, C. (2007) An ontology for software component matching, International Journal on Software Tools for Technology Transfer (STTT) (9:2), pp 169-178.
Phelps, S (2007) Evolutionary Mechanism Design, Ph.D. dissertation, University of Liverpool.
Rao, J., Kungas, P. and M. Matskin (2004). Logic-based web services composition: from service description to process model. Web services, IEEE International Conference on, pages 446-453.
Roman, D. (2005), Web Service Modeling Ontology, Applied Ontology (1:1), pp 77-106.
Sirin, E., Hendler, J., and Parsia, B. (2002), Semi-automatic composition of web services using semantic descriptions, Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003.
Stollberg, M. (2005), Reasoning Tasks and Mediation on Choreography and Orchestration in WSMO, Proceedings of the 2nd International WSMO Implementation Workshop, Innsbruck, Austria).
Streit, A. et al. (2005), UNICORE - From Project Results to Production Grids, L. Grandinetti (Edt.), Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing 14, Elsevier,), pp 357-376.
Sycara, K., Paolucci, M., Ankolekar, A., and Srinivasan, N. (2003), Automated discovery, interaction and composition of semantic web services, Journal of Web Semantics (1:1), pp 27-46.
Tenorio, R. (1999), Multiple unit auctions with strategic price-quantity decisions, Economic Theory, Springer, vol. 13(1), pages 247-260.
Wurman, Peter R. (1999), Market Structure and Multidimensional Auction Design for Computational Economies, Ph.D. dissertation, University of Michigan, August.