2008

Provisioning of Service Mashup Topologies

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

Wibke Michalk
Information Management and Systems, University of Karlsruhe, Germany, michalk@iism.uni-karlsruhe.de

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

Christof Weinhardt
Information Management and Systems, University of Karlsruhe, weinhardt@iism.uni-karlsruhe.de

PROVISIONING OF SERVICE MASHUP TOPOLOGIES

Blau, Benjamin, Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany, blau@iism.uni-karlsruhe.de

Michalk, Wibke, Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany, michalk@iism.uni-karlsruhe.de

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

Weinhardt, Christof, Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany, weinhardt@iism.uni-karlsruhe.de

Abstract

Service-orientation has gained great momentum in technological and economical matter. This development forces companies to drive new strategies such as service differentiation, coopetition and engaging into dynamic service ecosystems. Traditional ideas of static value chains give way to agile service value networks where many decentralized service providers compete and cooperate in the creation of value added mashed-up services. Nevertheless, there is a lack of efficient ways to design service mashups considering service configurations, interrelations and incompatibilities. There is a necessity to foster flexible allocation of sub-services and dynamic pricing. Currently, services are mostly charged statically via flat fees or pay-per-use prices. This is controversial because especially in short-living environments as service markets, there is a great demand for mechanisms providing dynamic sub-service reallocation and overall price determination.

We present an ontology framework which is integrated into a tool that supports service intermediaries in the design process of service networks. These networks represent possible instantiations of a value added mashed-up services formed by various decentralized service providers. Based on this, we provide an application of a path auction with a Vickrey-Clarke-Groves and a Low-Payment-Mechanism to determine the price of mashups. We evaluate proposed mechanisms exploring different graph topologies and bid strategies.

Keywords: Service Mashup, Service Value Network, Pricing, Path Auction, Mechanism Design

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 community-oriented purpose like Flickr (http://flickr.com) or del.icio.us (http://del.icio.us). 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 (http://Xignite.com) or Reuters which offer finance services for a large number of different users. Another example is StrikeIron (http://strikeiron.com) which provides Web services in the field of CRM, Business Intelligence and data cleansing and processing.

Nevertheless, there is still a big lack of efficient ways to design consistent service mashups, dynamically allocate potential sub-services and enable dynamic pricing. Currently, services are mostly charged flat fees and pay-per-use prices. Static pricing is the most common pricing scheme used. This fact is controversial because especially in short-living, highly-adaptable environments such as service markets, there is a great demand for mechanisms providing dynamic sub-service reallocation and overall price determination.

This paper bridges the gap between a technical/semantic view on the challenge of service provisioning and an economic perspective with respect to service allocation and price determination. The contribution of our paper is threefold:

We introduce an ontology framework that is part of a tool which visually and semantically supports service providers in the process of service mashup planning. Our ontology-based framework for modeling services provides concepts for specifying functional and non-functional service properties with a special focus on economic aspects. The result of the planning process is a graph topology representing the complex service and its potential sub-services, their configurations and reasonable interrelations that fulfill an overall functionality.

This graph is the basis for our second contribution which is the application of a path auction with a VCG and a Low Payment Mechanism that facilitates dynamic price determination of service mashups. The mechanisms are described formally, incentives to participate in the auctions are shown and the mechanisms' assets and drawbacks are presented.

In order to substantiate the theoretical results, we provide a comprehensive evaluation by using numerical simulations. We show that the Low Payment mechanism is at least as suitable for dynamic pricing of service mashups as the VCG mechanism independent of the network's topology or agents' strategies. As well, the Low Payment mechanism yields better prices than the VCG mechanism despite the agents placing bids that are higher than their true valuations.

This paper is structured in six sections. Section 2 motivates the need for supported planning and dynamic pricing of service mashups. We introduce a portfolio optimization service as concrete service mashup scenario from the finance domain. In section 3, we present an ontology framework that supports the planning process and works as a design pattern for service mashup composition. The framework enables modeling economic and non-functional service descriptions. The mechanisms used for dynamic price determination in service mashups are presented and evaluated in section 4. In section 5, we analyze the mechanisms by simulating networks of different size and degree of connectivity and by agents choosing different strategies. Section 6 closes with a conclusion and points out further challenges and future work.

2 SCENARIO

The scenario consists of a service provider who wants to offer a service mashup that optimizes a given portfolio. This finance service is based on a bundle of sub-services retrieved from decentralized sub-service providers (Figure 1). Suppose there are several sub-service providers which are competitors for each type of service such as storage, content and computing.

In order to provide the functionality of optimizing customers' portfolios, the service facilitates various sub-functionalities like storing, finance data provisioning, and computing. In our scenario, storage is provided by either Amazon S3 (Simple Storage Service) or Google's GDrive. Finance information can be obtained by either Reuters Online or Xignite which offer services that provide historical and real-time finance data such as stock quotes, currency rates or various indices. The Reuters service provides data concerning global news, company fundamentals data, company reports, stock prices, bonds, currencies and commodities. Xignite Web services provide market data (stock quotes, bonds, currencies) as well as corporate and industry data. In order to run optimization algorithms the service intensively requires computing power which is offered by either Sun Grid Computing Utility (network.com) or Amazon Elastic Compute Cloud (EC2).

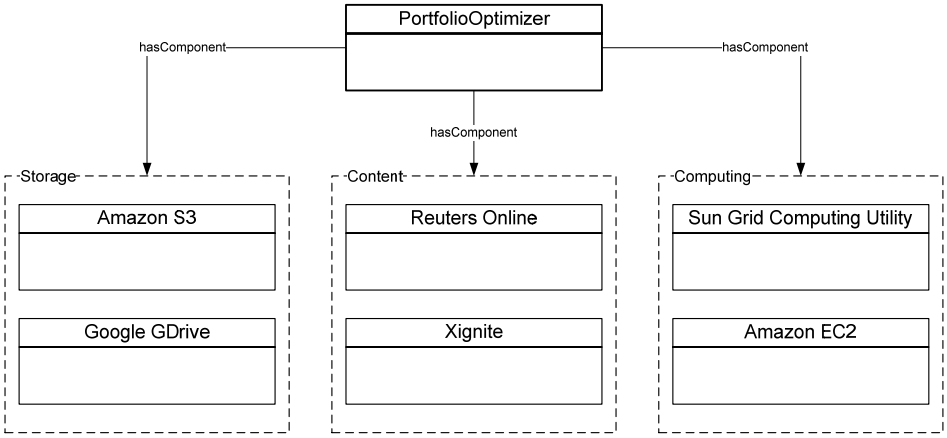

Figure 1 Service Mashup Scenario

3 PLANNING OF SERVICE MASHUPS

The process of planning complex highly configurable services like portfolio optimization is very sensitive and poses many challenges. Service providers face issues like sub-service incompatibility from a technical and contextual (semantic) perspective. Furthermore, there are various interdependencies between different configurations of sub-services. To sum up, in every service domain, there exists vast expert knowledge on service bundles and sub-service interrelations which as to be integrated in a system that is capable of supporting service providers in the service mashup planning process.
In this section, we propose a three-layered ontology framework which meets stated requirements and proposes a solution for tool-supported planning of services.

3.1 Ontology Framework

As a formalism to represent our ontology framework, we use OWL. OWL is an ontology language standardized by the World Wide Web Consortium (W3C) (W3C, 2004) and is based on the description logic (DL) formalism (Baader, Calvanese and McGuinness, 2007). Due to its close connection to DL, it facilitates logical inference and allows to draw conclusions from an ontology that have not been stated explicitly.
We suggest an ontology framework for modeling service mashups which is structured in three parts: The *Generic Service Ontology*, the *Domain Specific Service Ontology*, and the *Service Instance Layer*.

3.1.1 The Generic Service Ontology

The first part represents the *Generic Service Ontology*, which defines relevant concepts to specify services in general and which is independent from a concrete application scenario (Figure 2).

The Generic Service Ontology functions as an ontology design pattern (Devedzic, 1999; Gangemi, 2005) for modeling service mashups in a standardized manner. Furthermore, the Generic Service Ontology is structured in three parts: *Service Context*, *Service Economy* and *Service Configuration*.
Concepts in the service context part support the description of contextual information on service interrelations. E.g. a service can function as a predecessor referring to another service. At the same time, it can be an enhancement for this particular service. In order to assure the correctness of service interrelations, the contextual information is supported by a set of rules which is explained in the Section 3.2.

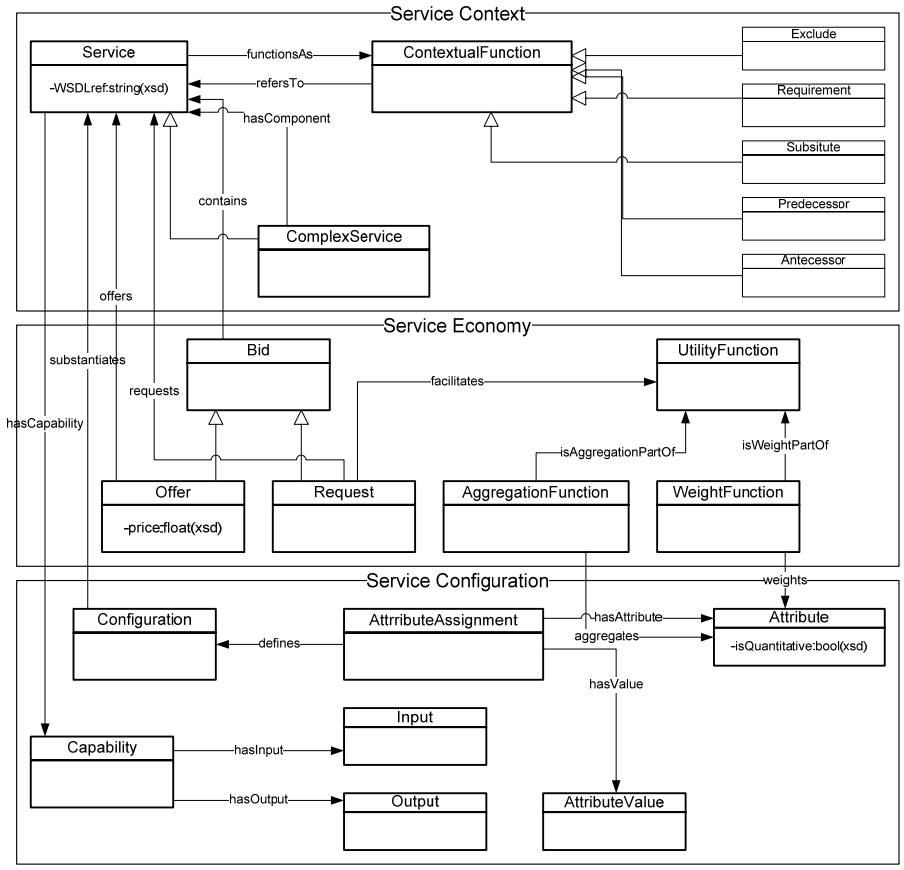

Figure 2: Generic Service Ontology

In the second layer of the Generic Service Ontology, concepts specify economical aspects of a service. A *Bid* can either be an *Offer* that specifies a traded service with a corresponding *price* or a *Request* for a certain service. For each type of attribute, the concept *AggregationFunction* defines a function that is capable of aggregating the attribute's value (e.g. response time is aggregated additively). Concepts for aggregating service properties have been discussed in detail in (Agarwal and Lamparter, 2005). The concept *WeightFunction* represents weights for different types of attributes which implicitly shows the requestors preferences.
The third layer represents the service configuration part that defines concepts for describing service capabilities and instance configurations. A *Configuration* is defined by an assignment of *Attributes* and their *AttributeValues* which form concrete service instances. The attribute concept specifies the semantic type of service property (e.g. response time, encryption mode) which can either be quantitative or qualitative. The *Capability* concept is divided in an *Input* and *Output* part.

3.1.2 The Domain Specific Service Ontology

The *Domain Specific Service Ontology* in the second part represents domain specific knowledge on the service mashup and its sub-services for a given scenario as depicted in Figure 3.
As service related concepts are subclasses of the central *Service* concept in the Generic Service Ontology, they inherit predefined concepts and relations supporting the modeling of service characteristics. The second part represents expert knowledge on a specified domain. This is realized with the aid of a concept called *ValidComplexService* which is a subclass of the *ComplexService* concept in the Generic Service Ontology. This concept is defined by a set of axioms representing domain specific knowledge with regard to service compatibility and interrelation issues of concrete service configurations.

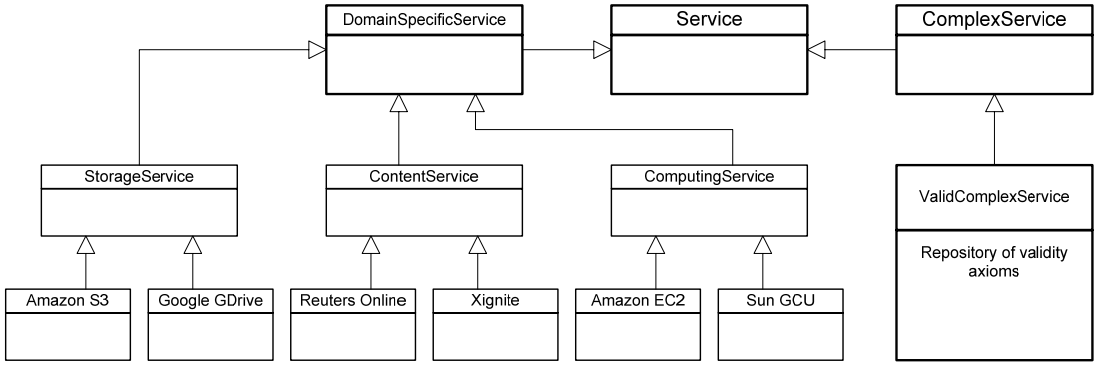

Figure 3: Domain Specific Service Ontology

3.1.3 The Service Instance Layer

The third part of the ontology is the *Service Instance Layer*. It shows the result of the planning process as an instance of the *ComplexService* concept. This instance describes a concrete service mashup, its possible sub-service configurations and their interrelation which is the subject matter of validation (Section 3.2.). The result of the planning process is a directed graph that represents all reasonable sub-services and their interrelations. This graph is the starting point for mechanisms to determine the overall price of the service mashup discussed in Section 4.

3.2 Planning Support

In order to support service providers in the planning process, our Ontology Framework respectively our tool provides model validation functionality for different design aspects.
The central supporting component for model validation is part of the Domain Specific Service Ontology. For validation, the model of the service mashup in the Service Instance Layer is added as an instance of the concept *ComplexService* to the Domain Specific Service Ontology. Then, a reasoning task is performed which checks if it can be inferred that the mashup is also an instance of the concept *ValidComplexService*. The concept *ValidComplexService* is specified by a set of DL axioms that restrict the set of valid service configuration bundles by defining constraints.
The following example shows an axiom defining the concept *ValidComplexService*. If an instance of the concept *AmazonEC2* functions as a predecessor referring to an instance of the concept *AmazonS3*, its configuration must have an attribute *AmazonUserID*. Instances that satisfy this axiom are inferred to be an instance of the concept *ValidComplexService*.

$$ValidComplexService \sqsubseteq ComplexService$$

$$ValidComplexService \sqsubseteq (\forall\, hasComponent\ (AmazonEC2 \sqcap (\exists\, functionsAs\ (Predecessor \sqcap (\exists\, refersTo.AmazonS3))) \sqcap (\exists\, isSubstantiatedBy\ (Configuration \sqcap (\exists\, isDefinedBy\ (AttributeAssignment \sqcap (\exists\, hasAttribute\ AmazonUserID \sqcap$$

$\exists\, hasValue.\{UserID\})))))))$

Another supporting component is specified by the concept *ContextualFunction* and more concrete sub-concepts like *Exclude*, *Requirement*, *Substitute*, *Predecessor* and *Antecessor*. These concepts and their meaning are defined by set of rules that assure that once a contextual relation between services is defined, the semantic implication is constituted and can be inferred by a reasoning task.
For the declarative formulation of matching directives in form of rules, we require additional modeling primitives not provided by OWL. We use the Semantic Web Rule Language (SWRL) (Horrocks, Patel-Schneider, Boley, Tabet, Grosof and Dean, 2004) which allows us to combine rule approaches with OWL. We restrict ourselves to a fragment of SWRL called DL-*safe* rules (Motik, Sattler and Studer, 2005) which is more relevant for practical applications due to its tractability and support by inference engines such as Pellet (Sirin, Parsia, Grau, Kalyanpur and Katz, 2007).
For example, a service B functions as a requirement referring to a service A and service A is part of a complex service C. This implies that service B must also be a component of complex service C to assure a flawless execution. These contextual rules also work vice versa.

$$Service(?x) \sqcap Service(?y) \sqcap ComplexService(?z) \sqcap hasComponent(?z,?x) \sqcap Requirement(?r) \sqcap functionsAs(?x,?r) \sqcap refersTo(?r,?y) \longrightarrow hasComponent(?z,?y)$$

If, for example, capabilities of a service A are subsumed by capabilities of a service B, service B functions as a substitute for service A.

$$Service(?x) \sqcap Service(?y) \sqcap hasCapability(?x,?a) \sqcap hasCapability(?y,?b) \sqcap subsumes(?a,?b)^{1} \longrightarrow Substitute(?s) \sqcap functionsAs(?x,?s) \sqcap refersTo(?s,?y)$$

In summary, we proposed an ontology framework that provides capabilities for expressive service descriptions including service input and output capabilities, configurations, economic aspects as well as dependencies and contextual information on service interrelations. Furthermore, we presented a validation concept that assures consistency of designed service mashups and a rule-based modeling support for contextual service functionality. The result of the planning process is a directed graph that represents all reasonable sub-services and their interrelations (Figure 4).

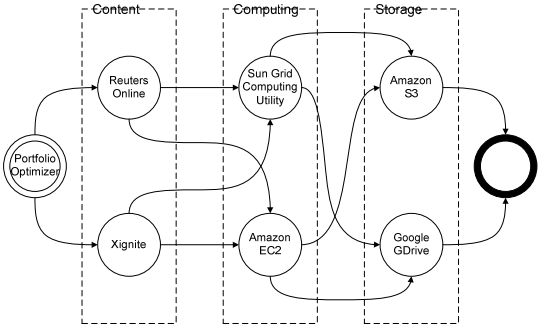

Figure 4: The result of the planning phase represented by a network graph

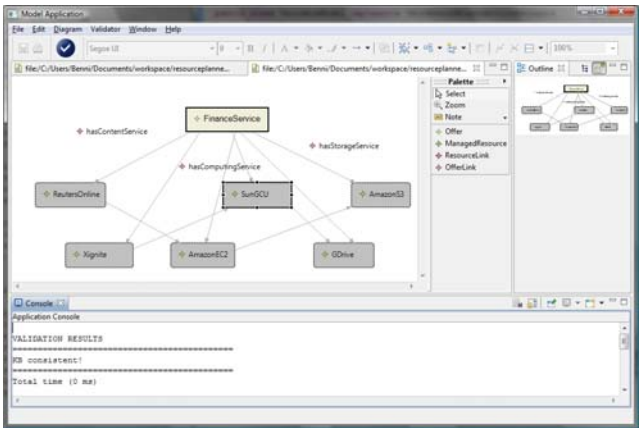

Figure 5: Service Mashup Planner

[1] *subsumes* is a SWRL built-in that verifies if a concept A subsumes a concept B meaning that the interpretation of concept B is a subset of the interpretation of concept A

The proposed concept is part of a tool called Service Mashup Planner[2] (Figure 5) which supports service providers in the configuration and planning process. The tool's architecture consists of the Eclipse Rich Client Platform, the Graphical Modeling Framework (GMF) and the Eclipse Modeling Framework (EMF). The ontological knowledge is integrated via the Jena Semantic Web Framework and processed by the Pellet Reasoner. The tool provides graphical modeling capabilities and comprehensive ontology-based validation functionality (cp. Section 3.2).

4 PRICING MECHANISMS FOR SERVICE MASHUPS

This section provides an overview about applying path auctions to the price determination of service mashups. Therefore, a Vickrey-Clarke-Groves mechanism that is working on the network graph resulting from the planning step is implemented. This mechanism incentivizes agents to reveal their true costs and thus, the resulting path will be the true lowest cost path (LCP). Another kind of path auctions is presented by the implementation of a Low Payment mechanism.

4.1 Motivation

The planning phase enabled a service provider to identify possible subservice configurations and their incompatibilities. Following the functional matching of subservices an economic composition has to be achieved for being able to determine a total price for the service mashup.
By now, most Web service providers charge the user flat fees independent of the extent of resource employment. Exemplarily, calling Sun's Grid Compute Utility[3] causes expenses of $1 per CPU-hr disregarding the amount of data transfer or the processor load. Amazon's pricing scheme for EC2[4] on the other hand allows for technical properties of the used instances and the extent of transferred data. This results in usage-driven prices satisfying the need for just paying what is used in an on-demand manner. However, this price determination is very complex and would especially in the case of service mashups lead to intricate price computations.
Throughout it is sensible to focus on prices that are determined in consideration of the way the service is used. Thus, we will introduce a descending path auction. Path auctions have recently been discussed in Archer and Tardos, 2007; Feldman et al. 2005; Ronen and Talisman, 2005. However, these path auctions have mostly been applied to network routing. Our scope will be the dynamic pricing of service mashups which is different to network routing with respect to service configurations and economic aspects.
Our aim is to find a way of minimizing prices brought to the requester's account and in the meantime incentivize subservice providers to participate in the complex service.
The reasons for using a path auction are:

- Different costs depending on antecedent services
- Different configuration possibilities
- Avoiding situations where only parts of the service mashup are purchased.

Using one auction per service would neither reflect incompatibilities nor differing costs depending on the antecedent service.
We expect agents to behave rational with respect to maximizing their quasi-linear utility. In addition, we assume that no monopoly situation exists where one agent is part of all feasible outcomes. We also assume that the topology is basic knowledge to all agents.

[2] http://www.iw.uni-karlsruhe.de/smp
[3] http://network.com
[4] http http://aws.amazon.com/ec2

4.2 Vickrey-Clarke-Groves

First, a mechanism implementing incentive compatibility is presented, where agents reveal their true costs as bids. Consequently, the combination of subservices can be found that yields the lowest price for the service mashup. Agents that are part of the LCP will receive a payment (bonus) in addition to the compensation of their costs. The resulting path price exceeds the costs of the LCP.

The result of the service mashup planning phase depicted in Figure 4, is represented by a directed disjoint graph $G = (V, E)$ with a start node v_s and a final node v_f. Each node v_j in G represents a sub-service owned by an independent service provider $s \in S$. Each edge $e_{ij} \in E$ forms the call of sub-service j by sub-service i. In order to reflect the fact that prices for services differ depending on the antecessor, the costs c_{ij} associated with each incoming edge $e_{ij} \in E$ to node j can be different. These costs are private knowledge, topology and edge-ownership are public knowledge.

Let F denote the set of feasible outputs, namely all paths from source to sink, where f_k is the k^{th} path from v_s to v_f represented by the set of edges connecting the nodes on f_k. As these paths clearly are a result of the planning phase, they can be used for the mechanism definition.

The mechanism can be described as $m = (o(b_i), p_i(o, b))$, where $o(b_i)$ is the allocation algorithm and $p_i(o, b)$ the payment to agent i and b_i represents the bid placed by agent i. The allocation to the path $f_k \in F$ incurring the lowest prices to the requester (Lowest-Cost-Path, LCP) is done by maximizing the social welfare what is equivalent to minimizing the sum over all agents' bids $b_{ij}(c_{ij})$ on a path:

$$\boldsymbol{o = argmin_{f_k \in F} p_{fk},}$$

where $p_{fk} = \sum_{e_{ij} \in f_k} b_{ij}(c_{ij})$. The bids $b_{ij}(c_{ij})$ depend on the costs incurring on the agents when executing the service. To incentivize agents to place their costs c_{ij} as bid $b_{ij}(c_{ij})$ in the descending auction, an agent i will receive a bonus payment $p_i(t)$ if one of its edges is on the LCP. Let $I^{ij}(LCP)$ denote the indication function for the LCP, where $I^{ij}(LCP) = f(x) = \begin{cases} 1, & e_{ij} \in LCP \\ 0, & otherwise \end{cases}$

To calculate $p_j(t)$, the Lowest j-avoiding path is computed and the costs of the true LCP minus $b_{ij}(i, j) \in LCP$ are subtracted:

$$\boldsymbol{p_j = \sum_{i \in j_{in}} I^{ij}(LCP(G', F') - LCP + b_{ij}),}$$

where F' is the set of paths through $G' = G \setminus \{j\}$, j_{in} equals the set of incoming edges to j and b_{ij} denotes the bid placed by node j for the costs of a service called by agent i. Obviously, if no edge of agent j is on the LCP, the payment $p_j(t) = 0$, what is tantamount to no payment when the agent does not execute a service. This payment is equal to a payment in a descending Vickrey-auction.

We expect each agent j to act rational and self-interested and so seek to maximize her quasi-linear utility $u_j = p_j(t) + \sum I^{ij}(LCP)\ (-c_{ij})$. An agent j's utility equals her profit from performing the service, so she might be better off not performing a job than performing it for a lower price than her real costs ($u_j < 0$). When bidding a higher than her true costs, she runs the risk of not being part of the LCP.

Thus, we introduced an individual rational VCG mechanism (Nisan and Ronen, 2001) that leaves all the agents i with a utility $u_j \geq 0$ and that induces truth-telling as dominant strategy, as stated in Feigenbaum, Papadimitriou, Sami and Shenker, 2005, so that $b_{ij}(c_{ij}) = c_{ij}$ is j's best strategy to place a bid.

With respect to the network graph resulting from the planning phase as in Figure 4, the mechanism is conducted. Assume the agents' bids are as follows: Accessing Reuters data costs \$12. The following computation job can be performed by using Sun's network.com 8 CPU/hours for \$8 or for \$9 by Amazon. As a storage server, Amazon's S3 can be used which costs \$2 if invoked by network.com or \$3 by Amazon. The set of bids is depicted in Figure 6.

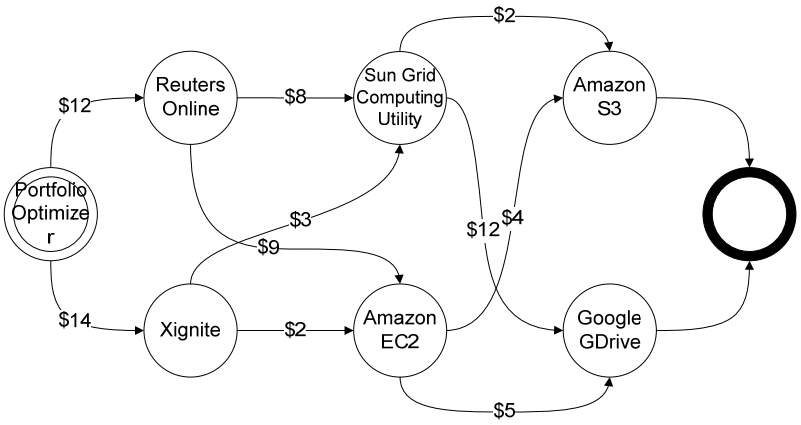

Figure 6: Service mashup graph with placed bids in a VCG mechanism

The LCP is computed using a well-established shortest path algorithm like Dijkstra. Therefore, the winning path is the use of Xignite, Sun and Amazon S3 with a total cost of \$19. The price the requester will have to pay is calculated as the sum of payments to all agents on the LCP: $c_{total} = \sum_{j \in LCP} p_j$. The Xignite service's costs are \$14. Without Xignite in the graph, the costs of the LCP amount to \$22 using Reuters, Sun and Amazon S3. Thus, the Vickrey price for Xignite $p_{Xignite} = \$22 - \$19 + \$14 = \$17$ Analogous, the price for Sun is calculated to $p_{Sun} = \$20 - \$19 + \$3 = \$4$ and Amazon S3's price to $p_{S3} = \$21 - \$19 + \$2 = \$4$. The total price to be paid for the service amounts to $p_{total} = \$17 + \$4 + \$6 = \$25$.

The VCG mechanism's best feature is to induce truth telling as dominant strategy. Due to the incentive compatibility of the mechanism, the service always is allocated to the path with the true lowest costs. The sacrifice the service requester has to make in return is that of a price that can become much higher than the costs incurring to the service suppliers as stated in Archer and Tardos, 2007. Therefore, we recommend the service requestor to set a reserve price. If the service price exceeds the reserve price, the requestor will not purchase the service.

Thus, we consider the VCG mechanism as a good starting point for pricing service mashups dynamically. As the price to be paid in incentive compatible auctions can become undesirably high, we introduce a descending First-Price auction that is not incentive compatible.

4.3 Low Payment Mechanism

Another auction mechanism for pricing in service mashups is the Low Payment mechanism as introduced by Immorlica, Karger, Nikolova and Sami, 2005; Ronen and Talisman, 2005. A First Price Sealed Bid (FPSB) auction is used for the determination of prices. Consider the same setup as in Section 4.2 with nodes v_f, the source v_s and sink v_f, costs c_{ij} associated with edges $e_{ij} \in E$. The mechanism $m = (o(b_i), p_i(o, b))$ has a similar allocation algorithm as the VCG mechanism: The allocation to the path $f_k \in F$ incurring the lowest prices to the requester (Lowest-Cost-Path, LCP) is done by maximizing the social welfare what is equivalent to minimizing the sum over all agents' bids on a path:

$$\boldsymbol{o = argmin_{f_k \in F} p_{fk},}$$

where $p_{fk} = \sum_{e_{ij} \in f_k} b_{ij}(c_{ij}, o)$ .The mechanisms' difference becomes obvious when looking at payments and agents' bids. In a reverse FPSB auction, bidders are paid the amount of money equal to their bid

$$\boldsymbol{p_j = \sum_{i \in j_{in}} I^{ij} b_{ij}.}$$

The agents' resulting utility function is $u_j = p_j + \sum I^{ij}(LCP)\ (-c_{ij})$. As this mechanism is not incentive compatible, the bids will not equal the costs as in Section 4.2. Moreover, as there do not exist dominant strategies in FPSB auctions, each agent j will try to imitate the second best (lowest) valuation in order to place a bid just low enough to still win the auction in order to maximize her utility. As the mechanism does not induce truth telling as dominant strategy and therefore, bids on services are different from VCG bids, the price to pay for the service mashup will change accordingly. The bids on edges tend to be higher than the costs depending on the strategy the agent chooses. However, the prices incurring on the service requestor will be lower than the mashup prices in a VCG mechanism again depending on the chosen strategy. In addition, the service requestor can set a price cap, beyond which he is not willing to buy the path, thus better prices can be achieved. The service providers still have an incentive to participate in the auction as their bids mostly are higher than their valuations. So each agent i will have a utility $u_j > 0$.

The same example as in 4.2 run using a Low Payment mechanism could result in bids approximately 1.2 times higher than the true costs shown in Figure 6. The resulting shortest path as depicted in Figure 7 is composed of using Xignite, Sun and Amazon S3 which eventually is the same as in the VCG mechanism. Although the path is the same, the price to be paid for the service mashup is a different one. It is computed as the sum of the bids associated with the edges on the LCP: $\$17 + \$3 + \$2 = \$22$. The resulting path in this case is the true Lowest Cost Path – which is not always the case in a Low Payment mechanism depending on the subservice providers' strategies and the total price is lower than the one to pay for the LCP in a VCG mechanism. To show the impact of the network's topology, we implemented a simulation. The simulation results are discussed in the following section.

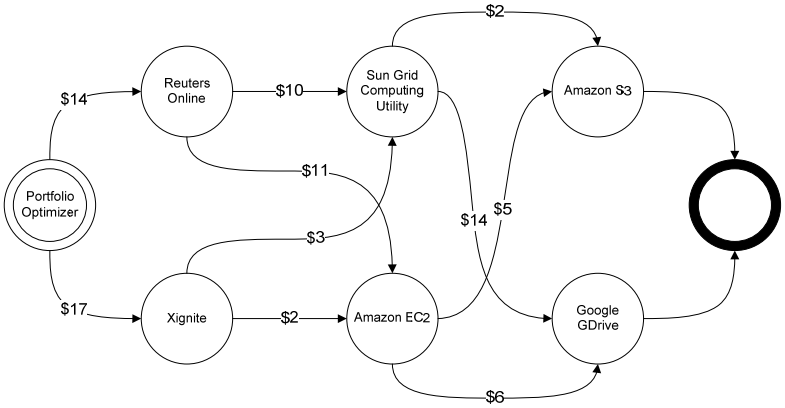

Figure 7: Service mashup graph with placed bids in a Low Payment Mechanism

5 SIMULATION AND EVALUATION

In order to illustrate the agents' behavior in service mashup networks, we implemented a simulation using Repast. A random network is created and the nodes act as agents placing bids on their incoming edges according to the mechanism currently used. Afterwards, the LCP and resulting prices are calculated. This section provides an overview on the effects on total prices for mashups and resulting utility for agents caused by changes in the topology or in manipulative behavior by single agents.

5.1 Impact of Network Topology

In previous examples as in 4.2 and 4.3 the underlying topology was fixed by a predefined network structure. Now, we create random networks with different numbers of agents to analyze the resulting prices, where prices are defined as $p = \sum all\ payments\ on\ the\ LCP$. The bids the agents place are drawn from a uniform distribution known to all agents as shown in Table 1. The degree of connectivity in the network, its density, changes. The question to be answered by this simulation is: Does the final price change dependent on the graph's density?

Attributes	Parameter Value
Number of nodes	50/100/200
Distribution of the interval (0,1)	Uniform
Density	0.1/0.2/0.3/0.4/0.5/0.6/0.7/0.8/0.9

Table 1: Parameters of networks simulated

Our results clearly show that prices decrease with increasing density. The higher the competition the lower the resulting prices – independent of the mechanism used for the auction. The exact findings are shown in Figure 8and Figure 9 where the density is depicted on the X-Axis, the path prices on the Y-Axis.

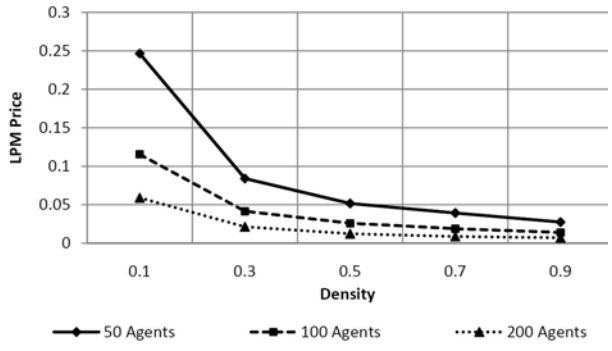

Figure 8: Mean path prices in LPM

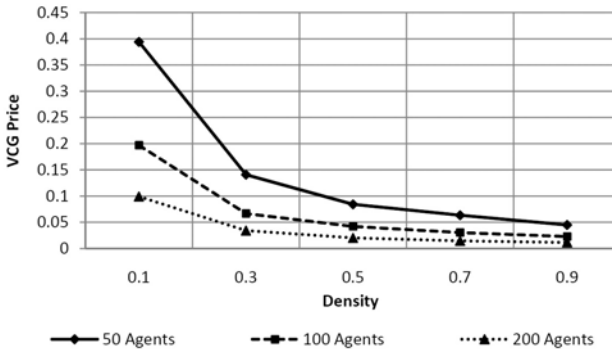

Figure 9: Mean path prices in VCG

5.2 Impact of Manipulating Agents

As a second step, we again create a random network and let one agent deviate from the respective optimal strategy. Exemplarily, in a VCG mechanism, one agent on the true LCP is chosen, the payment calculated and afterwards, his strategy is changed to bidding an amount slightly higher than her true valuation due to the respective deviation value. Subsequently, the new LCP is calculated and the resulting payment to the agent is computed. As the payment to the agent in VCG mechanisms does not depend on her bid but und the second best path's price, the agent will either receive a payment equal to zero (if not allocated) or the same payment as before. In the Low Payment Mechanism, the payment to the agent changes according to her bid and therefore, the difference between payments before and after deviating are compared.

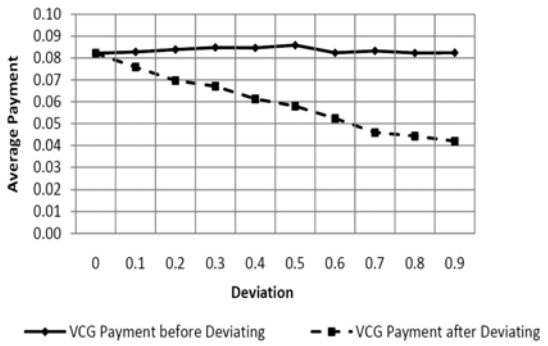

Figure 10: Mean Payment to Agents in a VCG

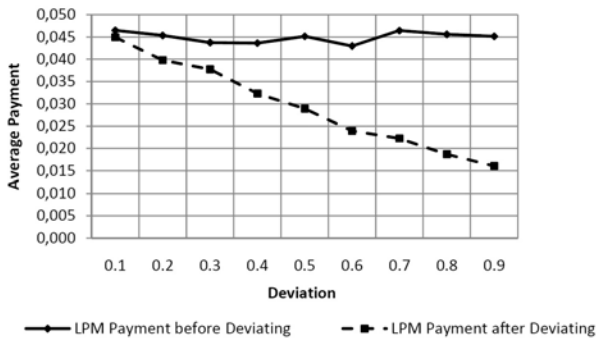

Figure 11: Mean Payment to Agents in an LPM

The outcome of the simulation disclosed that deviating from the optimal strategy never is useful for the agent neither in a VCG nor in a Low Payment Mechanism. The probability of being allocated in a VCG mechanism decreases with an increasing deviation as shown in Figure 10. Figure 11 shows that the mean payment to an agent decreases with increasing deviation in an LPM

6 CONCLUSION

In this paper, we proposed a way of configuring service mashups using an ontology framework. The configuration's result is a directed graph representing all feasible subservice combinations. Based on this graph, we introduced two different path auction mechanisms for price determination and discussed their suitability. Finally, we evaluated our mechanisms simulating different random scenarios to show the impact of the number of agents participating and the network's density on the resulting total price. As well, we showed the effect on one agent's payoff if she deviates from the mechanism's optimal strategy.

For future work, we plan to implement pricing mechanisms that consider more than just the service's price but as well attributes as the maximum time needed to perform the job, an encryption level and other service properties. We will analyze the case of agents concluding an SLA with the service requester. The resulting payoffs to agents and path prices will depend on the execution of the job to incentivize agents to stick to the SLA.

References

Agarwal, S. and Lamparter, S. (2005) *User Preference Based Automated Selection of Web Service Compositions,* ICSOC Workshop on Dynamic Web Processes, 1-12.

Archer, A. and Tardos, E. (2007) *Frugal Path Mechanisms,* ACM Trans. Algorithms, 3 (1), pp. 3.

Baader, F., Calvanese, D. and McGuinness, D. L. (2007) *The Description Logic Handbook: Theory, Implementation, and Applications,* Cambridge University Press.

Devedzic, V. (1999) *Ontologies: Borrowing from Software Patterns,* Intelligence, 10 (3), pp. 14-24.

Feigenbaum, J., Papadimitriou, C., Sami, R. and Shenker, S. (2005) *A BGP-based Mechanism for Lowest-cost Routing,* Distributed Computing, 18 (1), pp. 61-72.

Feldman, M., Chuang, J., Stoica, I. and Shenker, S. (2005) *Hidden-Action in Multi-Hop Routing,* ACM E-Commerce Conference (EC'05)Vancouver, British Columbia, Canada.

Gangemi, A. (2005) *Ontology Design Patterns for Semantic Web Content,* Proceedings ISWC 2005, 262-276.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2004) *SWRL: A Semantic Web Rule Language Combining OWL and RuleML,* W3C Member Submission, 21.

Immorlica, N., Karger, D., Nikolova, E. and Sami, R. (2005) *First-price Path Auctions,* Proceedings of the 6th ACM conference on Electronic commerce, 203-212.

Motik, B., Sattler, U. and Studer, R. (2005) *Query Answering for OWL-DL with Rules,* Web Semantics: Science, Services and Agents on the World Wide Web, 3 (1), pp. 41-60.

Nisan, N. and Ronen, A. (2001) *Algorithmic Mechanism Design,* Games and Economic Behavior, 35 (1-2), pp. 166-196.

Ronen, A. and Talisman, R. (2005) *Towards Generic Low Payment Mechanisms for Decentralized Task Allocation,* Proceedings of the 7th International IEEE Conference on E-Commerce Technology.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. and Katz, Y. (2007) *Pellet: A Practical OWL-DL Reasoner,* Web Semantics: Science, Services and Agents on the World Wide Web, 5 (2), pp. 51-53.

W3C (2004) *Web Ontology Language (OWL).*