Quantitative and Qualitative Description of the Consumer to Provider Relation in the Context of Utility Computing

Benjamin Heckmann[1], Andrew D. Phippen[2]
In memoriam Günter Turetschek

[1]h_da – University of Applied Sciences Darmstadt, Germany
[2]Centre for Security, Communications and Network Research
University of Plymouth, United Kingdom
benjamin.heckmann@gmx.de

Abstract: Utility Computing service provision aims to control the service quality for a wide range of consumers. To closely control the desired service quality in each phase of the service operations lifecycle, it is essential to be able to describe the quantitative and qualitative relation between consumer and provider. This work introduces Usage Patterns as a description language for the planning of quantitative relations and Provisioning Factors to control the qualitative relations during runtime.

1 Utility Computing

This work is focused on the modelling and simulation of service usage in the context of Utility Computing (UC). The term *utility* thereby refers to the field of industry. Here a public utility [Bri10] describes an enterprise that provides certain classes of services to a wide range of consumers.

The name Utility Computing indicates the vision of IT-based services comparable to public utilities. In this work Utility Computing is defined as a business model for service providers offering IT-based services and charging service consumers per usage, according to [Rap04]. From the provider's IT perspective UC is about service provision that is able to scale dynamically, according to real-time fluctuations in demand [BT06]. Additionally, UC service provision offers its services equipped with the ability to charge service consumption per use [Nee02].

From a consumer's perspective UC is related to "the reduction of IT-related operational costs and complexity" [YdAY+06]. Both perspectives, provision and consumption, have in common to target a better utilisation of generally underutilised IT resources [AAR02] on both sides. In summary, UC implicitly claims an abstract description of how IT resource utilisation, its total costs and service prices relate (see Figure 1).

Thereby Utility Computing does not refer to a specific IT service definition. From a business perspective any IT service where, from an economical point of view, it makes sense to charge it by its usage is addressed by UC, e.g. flight scheduling, webspace offers or

others. Therefore a more abstract service definition is the most suitable for UC: A service represents a type of relationship-based interaction between a service provider and a service consumer to achieve a certain solution objective [Zha07]. From a technical perspective there are several types of services that fit this definition, e.g. web services (SOAP, RESTful and others), HTTP web servers or virtual infrastructures (Xen, KVM and others). Service types outside the UC scope are e.g. IT projects or hardware sales.

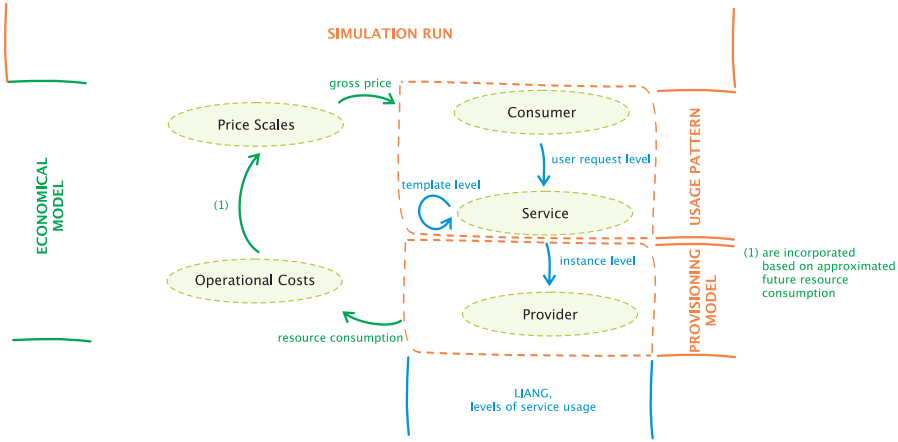

Figure 1: Levels of service usage in the context of this work

2 Service Quality

Utility Computing strongly addresses the aspect of service quality. This is implied by the vision of UC as a business model for IT service providers comparable to public utilities in case of necessity, reliability, usability, utilisation, scalability and exclusivity [Rap04]. All these attributes directly or indirectly address the quality of the service provision.

The definition about what quality criteria for a service offer are chosen and how they are monitored is subject to a service level agreement (SLA)[1] between the service provider and a service consumer. This work addresses SLA on the level of technical agreements of service quality, not the functional level. Thereby *technical* means: all technically measurable requirements relevant for the attended service response properties beside functional correctness. Functional correctness describes the accurate behaviour of a service on the layer of business logic. For example, a service method invocation with accurate method parameters, according to the specified parameter value ranges, must compute the accurate result set consisting of the expected business data.

[1] referring to the context of ITIL

In this work, response time is defined as the primary SLA criterion for the service consumer's perspective in the context of UC. Other possible primary SLA criteria, like continuity or security of service offers, are neglected. From the UC consumer perspective, request response times must be independent of the overall provider load. This implies the abdication of contracted resource reservations of any kind. Otherwise the targeted dynamic scale and optimised resource utilisation cannot be addressed by the UC provider. Response time is the outcome of the secondary SLA criteria on the service provider side, defined as request processing complexity and overall request amount. These criteria are subject to the individual SLA between provider and consumer. This definition differs from known definitions of SLA, which use resource centric criteria and therefore do not recommend for UC scenarios.

In addition to service quality criteria, according classes of observed criteria value ranges and corresponding actions must be specified during contracting. For the criterion of response time, aberrations could be classified as:

- Better – for service response times beyond the minimum specified acceptable period of response time.
- Within acceptable range – for service response times between the minimum and the maximum of specified acceptable periods of response time.
- Beyond unacceptable limit – for service response times beyond the maximum specified acceptable period of response time.
- Unprocessed requests – for technically and functionally accurate service requests that never got processed by the service provider, e.g. for dropped requests due to resource overload on the provider side.

Beside the definition of service quality criteria, the management of service quality is of interest to the service provider. A classical approach to manage the quality of service provision is the capacity management. Here the goal is to provide the necessary amount of resources for a certain service quality level at any time. This paper additionally proposes to manage the quality of service provision by managing the service usage. In this work *usage management* contains the indirect *consumption management* affecting the consumer side and the direct *provision management* on the provider side. Consumption management uses the provider's abilities such as SLA or price scales to indirectly influence the service consumer's usage behaviour. Therefore it addresses the quantitative aspect of the consumer-provider relation. For the qualitative aspect of the relation, provision management monitors, evaluates and directly controls service request routing and processing resource utilisation.

3 Research Objectives

The overall context of this work focuses on specific aspects of the service operations lifecycle (SOL) [HSPT09][2] for service offers based on the business model of Utility Computing. In the phase of service business planning this work refers to the corresponding service properties and service usage profiles resulting from the previous UC definition. During service development and the phase of service operations this work will focus on services in the technical context of Service-oriented Computing (SOC) [Pap03] corresponding to the paradigm of Cloud Computing as described by [BMQ+07].

In this context a description of the modifications necessary to transfer a standard service operations lifecycle into a UC SOL is missing. This includes the demand for an explicit definition of UC's core relation between IT resource utilisation, its total costs and service prices. Also specific attention must be given to the implications of complex UC usage scenarios.

The unidentified implications of complex UC usage scenarios considerably compromise the planning, development and operation of UC service offers. Under these conditions the prediction of resource utilisation and dependent operational costs, calculation of subsequent price scales, and subsequent runtime gross price calculations will fail.

4 Research Approach

The overall work starts from the business perspective, as technical requirements depend on the business requirements imposed. Therefore, a five step approach to find solutions for the specified objectives is proposed:

1. Describe the current state of service usage in the context of Utility Computing.
2. Elaborate a detailed definition for the relation between a service and its consumer.
3. Analyse the SOL of UC services.
4. Determine the implications of complex UC usage scenarios regarding SOL.
5. Deduct a corresponding strategy to handle the complexity.

This paper focuses on the elaboration of the quantitative and qualitative relation between service consumers and providers in the context of UC service provision quality management.

[2] SOL phases are defined as: business planning, development and operations

5 Usage Patterns

To be able to describe the core relation of Utility Computing, the following is assumed: UC implies a relation between IT resource utilisation, its total costs and service prices. This relation basically can be described by the quantitative usage relation between consumers and a service, enriched by metadata. All other variables are deducted from this usage relation, like cost and price calculations. This work proposes *Usage Patterns* as a description language for this quantitative usage relation.

For the IT architect's work, "new data, which includes usage patterns, will be added to the list of things to be considered" [Men07]. But the term Usage Pattern is not clearly defined in computer science. Some similar terms are used to describe traffic in computer networks or load in enterprise data centres. But none of these terms is applicable in the Utility Computing service provision context of this work.

As an entry point to service usage description the works of [LyCMO06] are introduced. Liang defines three perspectives of service usage and collects data on these levels as entry points for his usage data mining on web services. These levels of service usage are:

- User request level – The user request level of service usage addresses the outer view on composite services. This perspective focuses on how composite services are used by the consumer. This level is not aware of the optional complexity of service cascades or the diversity of providers within the cascade.
- Template level – The template level of service usage addresses the inner view on service correlations. A service template is defined as a flow of services, the final output of which can satisfy the consumer's need. At this level service usage concentrates on how services correlate.
- Instance level – The instance level of service usage addresses the constraints of the service runtime environment of the service provider. These constraints restrict how services are implemented and whether and how they can function.

Usage Patterns are concerned with the user request level and template level of Liang's definition of service usage, see Figure 1. Both of these perspectives describe relations between services and their consumers.

A Usage Pattern defines the quantitative usage relation between an unlimited number of service consumers and a particular service offered by a service provider. Thereby consumers are grouped according to their usage behaviour. This behaviour is expressed by one or more request classes, whereby the relation provides the request frequency as attribute with equal distribution assumed. Each request class describes a certain usage behaviour towards the function of a service.

This behaviour is described by an abstract function parameters class. Each class represents a characteristic combination of function parameter value ranges that imply a certain function call behaviour. It is assumed that for most functions the resource demand for processing a function call can be deducted from given parameter values. It is known that there are functions where this assumption fails, e.g. a function to calculate the total amount of a

bank account given the account number. It is not possible to estimate the resource demand of this calculation by evaluating the account number.

Also a request class may relate to any number of sub-request classes. For this recursive relation a request frequency attribute is provided, with equal distribution assumed. It is known that this ability to describe service cascades breaks the paradigm of service abstraction [Erl07]. Therefore this feature is optional.

The detailed relations which altogether instantiate a Usage Pattern are shown in Figure 2 using an entity relationship diagram [Che76].

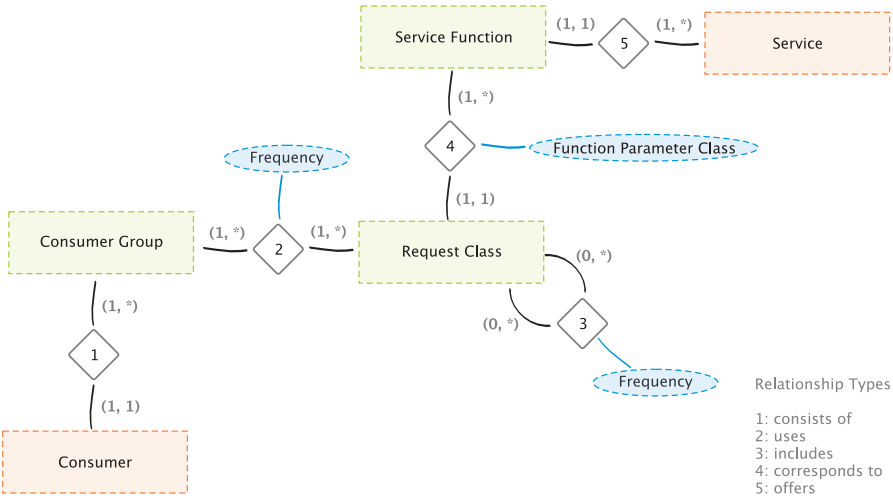

Figure 2: Usage Pattern as ER diagram

6 Service Operations Lifecycle Interaction

Imagine a service provider plans to offer a new web service [(W304] consistent to the business model of Utility Computing. In the first phase of the SOL, business planning for this service offer is conducted. In this example the executive expects three market segments in which the service could successfully be offered. In each segment different consumer numbers and usage behaviour are expected, due to the analyses of typical consumers. Based on these expectations a first simple Usage Pattern instance is derived.

Given this first pattern, IT architects and operations managers now jointly estimate the resource consumption of the future service architecture. This estimation is incorporated into the business planning by deducting the operational costs for the predicted resource consumption. This gives the executive the chance to validate planed price scales at an early stage. Beside the Usage Pattern instance representing the expected consumer behaviour, the executive elaborates a worst and a best-case scenario. In addition to the estimation

of the quantitative consumer behaviour, the executive specifies the service quality to be offered in each market segment.

As a base for their estimations IT architects will need a suitable provisioning model. Both will need a simulation of such a model to evaluate service quality and resource demand of service cascades. Both items, the model and its simulation, are described in [HSPT09].

In the development phase of the SOL the IT architect details the given Usage Pattern instances from the previous SOL phase. In this example it is estimated that the orchestration of another web service is reasonable. To accelerate the development reuse is chosen. Based on this extended Usage Pattern instance, conducting a simulation of the planned architecture helps the IT architect to validate previous resource demand estimations early. Even more importantly: he can validate service quality early and continuously during development.

After the new service is transferred to operations, the responsible manager needs to manage the service provision quality. Beside the indirect ability of consumption management, a continuous capacity planning is essential. Based on the estimation of future usage behaviour, provided by the executive, capacity demand for all offered services is simulated, respecting their interactions. This analysis is conducted as a worst-, standard- and best-case scenario to enable the executive to decide about future investments. Again, these quantitative conditions are described in Usage Patterns, thereby representing the base for simulation runs. Beside this continuous capacity planning in the SOL phase of service operations, service quality is guaranteed by provision management. Provision management ensures service quality by managing the routing of service requests. To calculate a routing decision, Provisioning Factors are used.

7 Provisioning Factors

The active management of service requests at runtime aims to gain direct control of the processing resource utilisation by controlling the routing of service requests. Besides the continuous monitoring of the utilisation of processing resources, the decision about the route of a request is the core of future provision management. To calculate this decision, measurable criteria, that both express technical and economical aspects of the request processing, must be defined. In this work these technical and economical criteria are called Provisioning Factors. These factors represent the qualitative aspect of the consumer-provider relation.

Provisioning Factors are segmented into three main factors:

- *Processing factor:* The processing factor aims to calculate the costs for the processing of a service request on provider-owned resources. These costs derive from the fixed costs for service hosting, e.g. for server acquisition, housing and administrative personnel, and corresponding dynamic costs, e.g. for cooling and power. The process to identify the individual combination of these fixed and dynamic costs is not part of this work. Before the calculation of the processing factor, resource

availability for request processing must be ensured. If the resources are available, the processing costs using the selected resources are calculated. This calculation includes all costs for sub-requests performed by the request. It is known that detailed analyses of large service cascades in order to find the optimum costs or to calculate the exact resource demand at runtime may fail in complex provisioning scenarios. In this case this work suggests calculating approximations instead.

- *Outsourcing factor:* Beside the option to process requests on provider-owned resources, scenarios are conceivable, where it can be an economical alternative to forward requests to other service providers for processing. Such outsourcing decisions can be appropriate for all layers of a service cascade. From processing customer requests on competitor sites in times of peak loads up to the dynamic processor picking for back-end services, such as the retrieval of geological information. The outsourcing factor aims to calculate the costs for external request processing.
- *Neglecting factor:* Instead of the two previous factors the neglecting factor aims to calculate the costs for an intentional violation of the SLA agreed with the consumer. Thereby the violation may at worst consist of a request drop, but also in other aberrations form the given SLA. To achieve this flexibility, the costs for all contracted variations of service level aberrations must be taken into account.

All three Provisioning Factors calculate costs. Combined with the consumer's contracted price list, the profit or loss of a request routing decision can be estimated. Note that all mentioned costs may vary over time on individually contracted factors, such as the time of day or discounts on request amounts. The introduced factors are only proposals used in this work. It is possible to add or remove criteria as needed in other contexts.

8 First Outcomes

First outcomes can be shown analysing an example scenario. In this example the profit of a service provider is analysed during peak demands, where significantly more resources to process all incoming service requests are necessary than are available. The example provider offers a single service to a certain range of consumers. The consumers can be grouped into three SLA groups. Each SLA group differs in maximum request response time, pricing and contractual penalty. It is assumed that in terms of request complexity, request frequency and request number each consumer behaves equally. The number of consumers in the highest and lowest SLA group is equal. The number of consumers in the medium SLA group is double the size of one of the other groups. The metered values in this scenario are the total number of requests and for each SLA group: number of request responses, mean of response duration, request drops, SLA fails and profit.

Compared are two scenario alternatives: classical vs. UC provision management during peak demands. In classical provision management resources are shared at a fixed ratio at runtime. During peak demands, this constraint also applies to more flexible classical resources sharing alternatives, where unused resources can be borrowed among other con-

sumers. For UC provision management it is assumed that request routing is adapted at runtime based on the Provisioning Factors introduced in this work.

The scenario is modelled as discrete-event model presenting a multi-tier IT architecture to process service requests. The model is implemented using the [HSPT09] UC simulation framework. Figure 3 shows the comparison between a classical and an UC simulation run. Both runs represent a simulation period of 30 minutes with 40 requesting consumers using random request invocations. Analysing the outcomes, the distribution and drop rates of requests between each SLA group are even. But there are significant improvements in means of response duration for the primary and secondary SLA group and in SLA fails for the primary SLA group. These shifts directly lead to a significant higher profit in the UC provision management scenario.

	Classic		UC		
[1] Total number of requests	15.594		15.738		
[2] Number of request responses					
Total	13.683	88% of [1]	13.704	87% of [1]	**Earnings per request**
SLA 1	3.539	26% of [2]	3.560	26% of [2]	0,25 €
SLA 2	6.963	51% of [2]	7.007	51% of [2]	0,05 €
SLA 3	3.181	23% of [2]	3.137	23% of [2]	0,01 €
[3] Mean of response duration					
Total	9,05 sec		10,46 sec		**SLA classes**
SLA 1	9,05 sec		7,40 sec		10,00 sec max.
SLA 2	9,05 sec		7,59 sec		15,00 sec max.
SLA 3	9,05 sec		16,38 sec		30,00 sec max.
[4] Request drops					
Total	1.911	12% of [1]	2.034	13% of [1]	**Drop fine per request**
SLA 1	573	30% of [4]	627	31% of [4]	0,50 €
SLA 2	1.129	59% of [4]	1.166	57% of [4]	0,02 €
SLA 3	209	11% of [4]	241	12% of [4]	none
[5] Request response SLA fails					
Total	1.729	11% of [1]	1.216	8% of [1]	**SLA fine per request**
SLA 1	981	57% of [5]	480	39% of [5]	0,25 €
SLA 2	726	42% of [5]	708	58% of [5]	0,01 €
SLA 3	22	1% of [5]	28	2% of [5]	none
[6] Provider profit					
Total	219,56 €		434,68 €	198% of classical [6]	
SLA 1	-35,50 €		179,75 €		
SLA 2	225,56 €		226,25 €		
SLA 3	29,50 €		28,68 €		

Figure 3: Analyses of a simulation run

9 Conclusions and Further Work

In this paper the quantitative and qualitative description of the consumer-to-provider relation in the context of Utility Computing is analysed. Derived from the demands of service quality management and the Utility Computing business model, Usage Patterns are introduced to address the definition of the quantitative consumer-to-provider relation. Usage Patterns are used to enrich the service operations lifecycle to enable the analyses of the qualitative consumer-to-provider relation during strategic planning. To address the qualitative consumer-to-provider relation at runtime, Provisioning Factors are introduced. Building on these factors, requests can be prioritised to optimise provider profits. The

introduced example uses a Usage Pattern to define its demand scenario. The simulation of the Usage Pattern shows that runtime request prioritisation raises profits, while higher service levels benefit from shorter response times. Further research aims to verify the simulation results analysing a real world scenario. The main aspect here will be the calibration of the simulation runs to reflect the current resource consumption of the simulated service requests.

References

[AAR02] Artur Andrzejak, Martin Arlitt, and Jerry Rolia. Bounding the Resource Savings of Utility Computing Models. 2002.

[BMQ+07] Greg Boss, Padma Malladi, Dennis Quan, Linda Legregni, and Harold Hall. Cloud Computing. *IBM DeveloperWorks*, October 2007.

[Bri10] Encyclopaedia Britannica. public utility. http://www.britannica.com, 2010.

[BT06] Guy Bunker and Darren Thomson. *Delivering Utility Computing: Business-driven IT Optimization*. John Wiley \& Sons, 2006.

[Che76] Peter Pin-Shan Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[Erl07] Thomas Erl. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR, 2007.

[HSPT09] Benjamin Heckmann, Ingo Stengel, Andy Phippen, and Guenter Turetschek. Utility Computing simulation. In *ESM'2009 The 2009 European Simulation and Modelling Conference*, pages 175–180, Leicester, United Kingdom, October 2009. EUROSIS-ETI.

[LyCMO06] Qianhui Liang, Jen yao Chung, Steven Miller, and Yang Ouyang. Service Pattern Discovery of Web Service Mining in Web Service Registry-Repository. In *2006 IEEE International Conference on e-Business Engineering (ICEBE'06)*, pages 286–293, Shanghai, China, 2006.

[Men07] Alfredo Mendoza. *Utility Computing Technologies, Standards, and Strategies*. Artech House Inc, April 2007.

[Nee02] Dan Neel. The utility computing promise. http://www.infoworld.com/d/networking/utility-computing-promise-807, April 2002.

[Pap03] M.P. Papazoglou. Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the 7th International Conference on Properties and Applications of Dielectric Materials (Cat. No.03CH37417)*, pages 3–12, Rome, Italy, 2003.

[Rap04] M. A. Rappa. The utility business model and the future of computing services. *IBM Syst. J.*, 43(1):32–42, 2004.

[(W304] World Wide Web Consortium (W3C). Web Services Architecture. http://www.w3.org/TR/ws-arch/, February 2004.

[YdAY+06] Chee Shin Yeo, Marcos Dias de Assuncao, Jia Yu, Anthony Sulistio, Srikumar Venugopal, Martin Placek, and Rajkumar Buyya. Utility Computing and Global Grids. *cs/0605056*, May 2006.

[Zha07] Liang-Jie Zhang. *Services Computing: Core Enabling Technology of the Modern Services Industry*. Tsinghua University Press ;;Springer, Beijing ;Berlin ;;New York, 2007.