Technical Report TR-CS-09-01

A Framework for Relating Frontstage and Backstage Quality in Virtualized Services

Karuna P Joshi, Anupam Joshi, Yelena Yesha
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250, USA

Ravi Kothari
IBM India Research Labs, Vasant Kunj,
New Delhi 110070, India

15 May 2009

COLLEGE OF ENGINEERING

UMBC

A Framework for Relating Frontstage and Backstage Quality in Virtualized Services

Karuna P Joshi[1], Anupam Joshi[1,2*], Ravi Kothari[2], Yelena Yesha[1]

1 Depart of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250, USA
{kjoshi1, joshi, yeyesha}@cs.umbc.edu

2 IBM India Research Labs
Vasant Kunj,
New Delhi 110070, India
{anupam.joshi, rkothari}@in.ibm.com

Abstract

Virtualized service models are now emerging and redefining the way Information technology is delivered to end users. In this paper, we propose a framework to measure and track quality delivered by a Virtualized service delivery system. The framework accounts for the service's internal elements as well as the other services it depends on for its performance. It provides a mechanism to relate hard metrics typically measured at the backstage of the delivery process to quality related hard and soft metrics tracked at the front stage where the end user interacts with the service. The framework is general enough to be applied to any type of IT service. In the paper, we show three instantiations of the framework – an IT enabled service, Software as a Service, and Infrastructure as a Service.

Keywords-component; Services; Quality Framework; frontstage, backstage

1. Introduction and Background

Businesses are increasingly relying on service providers for services that are crucial, but nonetheless outside their core competency. In some cases, the business may utilize multiple service providers to mitigate risks that may be associated with a single provider. In other cases, a business may utilize a single provider that in turn utilizes the services of other providers. In either case, the delivery of a service is often based on the composition of multiple other services and assets that may be supplied by one or more entities. The service, in effect, is virtualized on the cloud. This virtualized model of service delivery [1] allows easier customization, better utilization, greater responsiveness and is presently the preferred method to deliver services ranging from simple services such as helpdesk and backoffice functions to more complex services such as Infrastructure as a Service (IaaS). Indeed, the virtualized model of service delivery even extends to IT Enabled Services (ITeS) which typically also include a large human element.

The virtualized nature of service delivery brings about new challenges in ensuring the quality of service delivery. In particular, while it is easier to express metrics related to individual services that comprise the overall service, and the resources they use (such as storage, network, processing power etc.), it is the overall and composed service that is experienced by the client or the business at its front stage. It is this perceived quality at the front stage of a service that often acts as a differentiator in an otherwise commoditized service delivery environment. Service Level Agreements (SLAs) related to customer satisfaction are often a part of delivery contracts. Translating the metrics related to individual components of the service (which are in a sense backstage metrics) to the frontstage experienced by the client or business will therefore allow a service provider to differentiate from competitors offering a similar service. Some efforts have been directed at measuring quality in other environments; however, to the best of our knowledge there is no current standard for measuring IT service quality in virtualized environments.

SERVQUAL, for example, is a multi-item scale developed by Parasuraman et. al, to assess customer perceptions of service quality in service and retail businesses [2]. The scale is based on five dimensions of service quality, viz. Tangibility, Reliability, Responsiveness, Assurance and Empath***y*** (also abbreviated as RATER) that can be adapted to meet the demands of the particular kind of service setting under assessment. SERVQUAL represents service quality as the discrepancy between a customer's expectations for a service offering and the customer's perceptions of the service received, requiring respondents to answer questions about both their expectations and their perceptions. In other words, identical perceptions can still

[*] This work performed while the author was visiting IBM India Research Labs on leave from UMBC

lead to different quality judgments. This is intuitive – the same service provided at a discount chain and at an exclusive designer shop will lead to very different quality judgments. The use of perceived as opposed to actual service received makes the SERVQUAL measure an attitude measure that is related to, but not the same as, satisfaction.

Parsuraman et. al. [3] also proposed a multiple-item scale (E-S-QUAL) for measuring the service quality delivered by Web sites on which customers shop online. The basic E-S-QUAL consists of four dimensions: efficiency, fulfillment, system availability, and privacy. The second scale, E-RecS-QUAL, is salient only to customers who had non-routine encounters with the sites and contains three dimensions: responsiveness, compensation, and contact.

However, these existing quality measures are not appropriate for IT/ITeS. For one, they are mostly defined for Business-Client (B-C) systems, where the interaction is between a human agent in a physical facility representing the business and a consumer. In the virtualized service delivery model, the environment is Business-Virtualized service providers-Customer (B-V-C). So the end interaction might or might not involve a human being, and is almost never directly with the business whose customer is consuming the service. While some elements of RATER type models such as Responsiveness and Reliability have analogs in the IT domain, others such as empathy do not seem to have obvious analogs. Moreover, typically measures such as Customer Satisfaction (CSAT) are lagging. Existing work also makes no efforts to relate them in any (analytic) form with measures that can be made at the contact point with the customer, such as the average call handling time, waiting time, and such. These in turn are not related in the existing models to measurable parameters at the elements (other services, resources) that the orchestration of a service requires.

In other words, existing models are largely confined to the front stage, both in terms of the objectively measured variables as well as the perceived measures such as quality. Given that a service can be using other services, or resources, at the backend, it is evident that front stage metrics will depend on what happens at the back end (or at the other stages of the service orchestration). One key question, for which this paper proposes an answer, is how to relate the backend metrics related to resources and the services on which this service depends, to the quality metrics at the front end of the service? "Hard" Metrics are often available for resources and even some applications (such as response time, throughput, packet drop rates, transactions per second, etc), and most service providers have tools to measure such parameters.

If such relationships are found, they can provide two significant advantages. For one they help avoid the "winners curse" by making sure that the quality related SLAs agreed to with the customer can be met with the resources and services available. Many service delivery contracts are structured to provide a bonus if the SLAs are exceeded and a penalty if they are missed. Relating quality to performance can also help to see if increased investment in some element that makes up a service (another service or resource) can increase the quality perceived by the customer and lead to bonus or help avoid penalty. Second, they can help decide what new services can be provided from the given resources.

The key contribution of this paper is a new framework that enables the translation of backstage metrics to those at the frontstage. It captures the dependency of a service on others, or on backend applications and resources. Linguistic rules are then used to define how quality measures of a service at the frontstage relate to those of its resources or the other services it calls. Fuzzy Logic [7] is used to reason over such rules to move from the known hard metrics at the backstage to the soft metrics at the front.

Fuzzy sets provide us with the ability to classify elements into a continuous set using the concept of degree of membership. The characteristic membership function not only gives 0 or 1 for membership, but can also give values in between. For instance, instead of expecting an exact numeric measure of dependence between two services, we could use a description such as dependence is high. The relation of a dependence measure to a linguistic term such as high or low will be captured in the membership function.

2. Service Coupling and Dependency

It is generally easier to control the performance of an individual service which does not depend on others. Its performance quality can be measured, and correlated with the performance of resources upon which it depends. However, the virtualized service delivery model requires composition of services to deliver the overall service to the client. The interactions between the individual services, many of which may come from different sources, makes it harder to provide quality measures for it in terms of the quality and performance of the underlying services. Our framework addresses this issue by explicitly capturing and utilizing the interactions and dependencies.

Service Dependence or *Coupling* (C) is a measure that we propose to capture how dependent the service is on other services or resources for its delivery. It is similar to, but not the same as, the Coupling measure used in traditional software engineering to describe the interdependence between two software modules [6]. Loose coupling or a Low Dependency factor indicates that the Service provider does not have to depend on other

services or resources to complete delivery of its service. High Dependency Factor or tight coupling on the other hand indicates that successful delivery of other services or availability of resources is a prerequisite for the completion of a service.

We capture a linguistic description of the dependency – define it as high or moderately high or low, or so on... The degree of dependency or coupling could be directly elicited by the experts who have created the service. Another option is to mine the historical data to obtain the dependency relationships. Initial attempts at mining this on Infrastructure services have shown promise [8]. In either case though, the question of creating the fuzzy membership functions remains. We believe that this can also be mined from historical data, and are doing this in our ongoing work.

When the dependency is between a service and some resource it uses, coupling will essentially be a function of how often the resource is used. For instance, the dependence of a service on the network layer might be measured by how often it is making a socket call, or how much data it is transferring. The dependence of a database on compute partition will be determined by how much compute resources it needs from that partition, and so on.

For coupling between services, we can also build on the work related to module level coupling that has been extensively studied in Software Engineering (see for instance [4][5]). Existing literature defines several different types of coupling, which we adapt for the services domain.

When services are linked together, they exhibit Environmental coupling which is caused by calling and being called by other services. Data flow and Stamp coupling is caused by the parameters of the service interface. Unlike modules in a code base, services tend to be independent and largely self contained. So the Control Flow coupling has minimum influence on coupling in IT Services. Similarly, unless the services share state by altering some shared data repository, Global coupling will not be a significant factor either.

Fenton and Melton [5] propose the following metric as a measure of coupling between two components *x* and *y:*

$C(x,y) = i + n/(n+1)$ where,

n = number of interconnections between *x* and *y*, and

i = level of highest (worst) coupling type found between *x* and *y*.

The level of coupling type is based on the Myers classification and is assigned a numeric value [5]. The higher the value, the more coupled the services. Another well known formula in Software Engineering due to Dhama [6] defines how a module is coupled. This definition, like most others in software engineering, is a global one. It provides a measure of how tightly coupled the module is with the rest. We need in particular to define pairwise coupling between two services where one uses the other. So we adapt Dhama's metric to define coupling between services x and y using the formula,

Service Coupling $C(x,y) = 1/ (i + u + g + \mathbf{r})$

where,

i = in data parameters – data sent from calling service x to called service y

u = out data parameters – data sent from called service y to calling service x.

g = number of global variables used as data

r = number of times x calls y.

The lower this measure, the more tightly coupled the two services are. We can define fuzzy membership functions to map these measures into linguistic variables.

3. Proposed Framework

The framework that we propose for measuring the performance can be applied to every domain of IT services.

3.1. Service Elements

Services comprise of three key elements, the Agents or Human Beings providing the Service, actual software that encodes the service provided and other services/resources that the service depends on for its delivery. Since all the three elements contribute towards the quality of the service, performance failure in any of the element will result in poor service quality. A service might not have one of the elements, i.e. it may have no human element or no dependency on other services. Usually SLAs exist for each element of the service that measure the performance of the service.

3.2. Example for Software as a Service

To illustrate our framework for SaaS, we apply it to a Collaboration tool service which is provided to the consumers via the web. The service provides capability to its consumers to collaborate online allowing them to conduct meetings, simultaneously work on documents, chat as a group, etc.

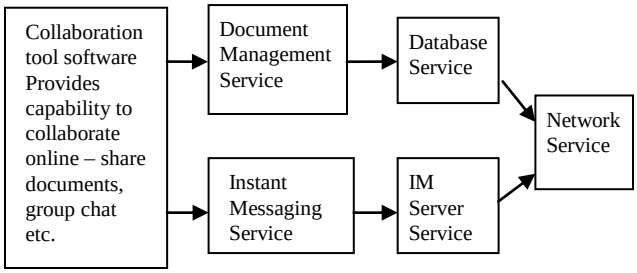


Figure 1: SaaS example - Collaboration tool services

As the service is completely automated, it has no service agents who provide the service. Hence, the service providers have a high degree of separation from the consumers who only contact them for technical assistance or if the tool performance is below par. The service is however highly dependent or tightly coupled with the underlying core services, like Databases and Network, for its successful delivery and this is illustrated in Figure 1.

The collaboration tool service is coupled with the following external services. Table 1 lists the main performance metrics used to measure the service's quality.

a. Collaboration tool is tightly coupled with Document Management service and Instant Messaging service.
b. Document Management service is tightly coupled with Database service.
c. Instant Messaging service is tightly coupled with Instant Messaging (IM) Server services
d. Database Service and IM Server Service is tightly coupled with Network Services

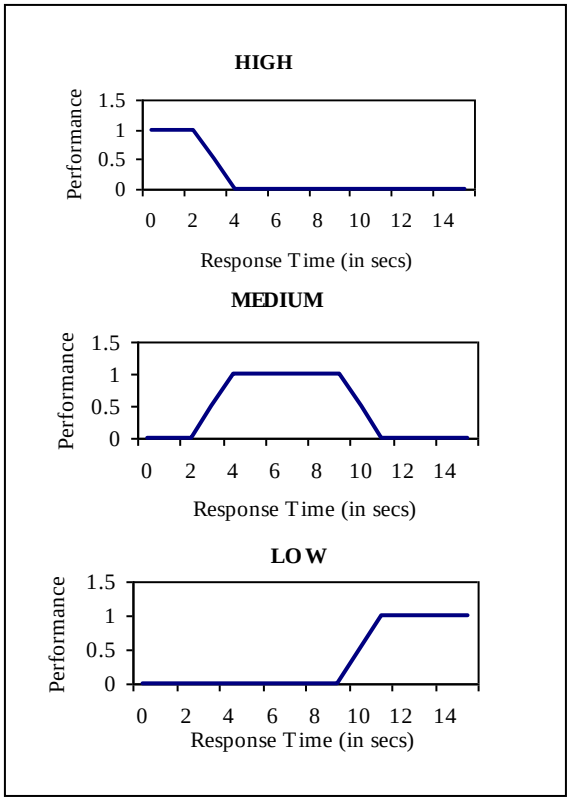
Table 1: Performance metrics used by Collaboration tool services

Metric	Measures what
Customer Satisfaction	Assessed through surveys of customers via telephone call, email or post.
Response Time	The time it takes for application to respond to user's (keystroke/mouse) commands.
Reliability of the tool	How consistently the software responds to user
Scalability	Performance of tool with increased number of collaborators
Number of Documents	Number of Documents users can collaborate on per session
Session	Average time spent by users on a session.

To measure the frontend quality of our service we apply fuzzy rules to the backend performance metrics which enable us to generate performance rules.

For this example, we fuzzify the "response time" performance metric into the fuzzy variables HIGH, LOW and MEDIUM. For the Collaboration tool service, Response time performance is LOW if response time is greater than 10 seconds. It is MEDIUM if response time lies between 3 and 10 seconds. It is HIGH if response time is less than 3 seconds. Graph 1 illustrates these fuzzy rules.

We have selected the response time metric as it can be applied across the main collaboration service as well as the services it depends on. The baseline for the various services will be different. Hence a response time of 2 seconds may point to HIGH performance for the Collaboration service, but it will be regarded as a LOW performance for Network Service.



Graph 1: Fuzzification rules for Response Time

Similar fuzzification rules can be applied to the other performance measures listed in table above. Once we have determined the fuzzy rules for our performance measures, we can create linguistic rules for the service that will help us determine the Service Quality. We are listing below some linguistic rules for the Collaboration Service

1. If {(Collaboration service tightly coupled with Document Management service) AND (Document Management service performance is LOW)} then the Collaboration service performance is LOW
2. If {(Document Management service tightly coupled with Database service) AND (Database service performance is LOW)} then the Document Management service performance is LOW.
3. If {(Database service tightly coupled with Network service) AND (Network service performance is LOW)} then the Database service performance is LOW.
4. If {(Collaboration service tightly coupled with Instant Messaging service) AND (Instant Messaging service performance is LOW)} then the Collaboration service performance is LOW

5. If {(Instant Messaging service tightly coupled with IM server service) AND (IM Server service performance is LOW)} then the Instant Messaging service performance is LOW
6. If {(IM Server service tightly coupled with Network service) AND (Network service performance is LOW)} then the IM Server service performance is LOW
7. If Collaboration service performance is LOW then Response Time is LOW
8. If Collaboration service performance is LOW then Reliability is LOW
9. {If Response Time is LOW OR Reliability is LOW} then Customer Satisfaction is LOW

Each of these performance rules should be evaluated for the service whose quality is being measured. Based on the rules, we can easily determine if the Service is performing at its desired level or not.

3.3. Example for IT enabled Service

We next apply this framework to a more complex IT service which consists of many more elements. Service Agent or the Human element of a service introduces more complex fuzzy rules.

To illustrate our framework for the IT enabled Services, we apply it to the IT Helpdesk application. This service provides technical solution/guidance to its consumers. Elements of Helpdesk service, illustrated in Figure 2, include Agents that provide the actual service of responding to the service consumers, software used to provide the service which is a CRM (Customer Relationship Management) application, and Automatic Call Distribution (ACD) software (integrated with PBX) used to automatically route the Helpdesk calls to the various agents.

Helpdesk service is coupled with the following external services. Table 2 lists the primary performance metrics used to measure the Helpdesk service quality.

a. Agent's expertise is coupled with the expertise of other Agents (i.e. Tier 2 helpdesk) or with external Agents (software experts etc.).
b. CRM software is tightly coupled with the Database server service.
c. The Database service is tightly coupled with the Network service.
d. The CRM software is coupled with the Knowledgebase service. The Knowledgebase could be a set of pre-determined solution list or FAQs or Help systems. Depending on the implementation it can be tightly or loosely coupled with the CRM application.
e. ACD+PBX software is tightly coupled with the underlying Telecommunication service.

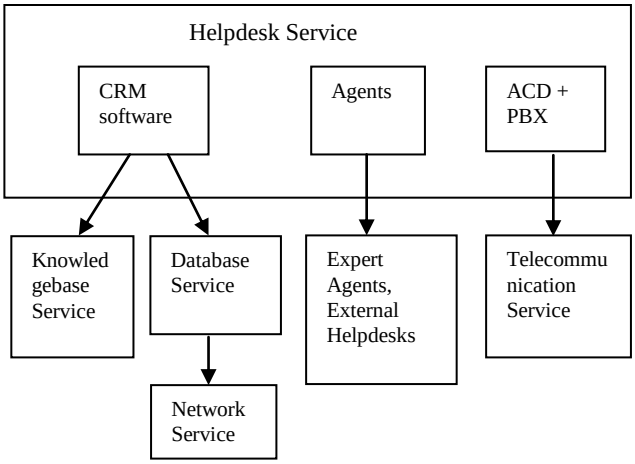


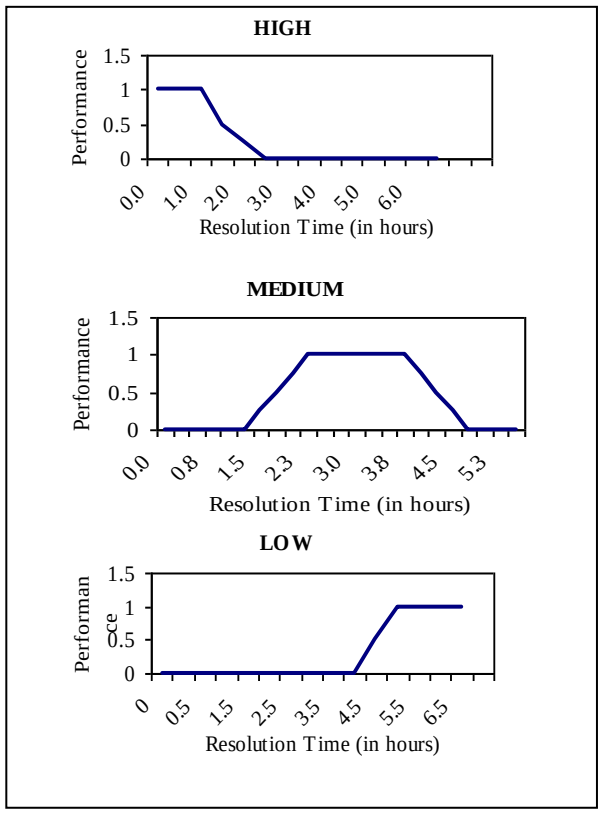
Figure 2: ITeS example: Helpdesk service

Table 2: Performance metrics used by Helpdesk services

Metric	Measures what	Helpdesk Element
Customer Satisfaction	Assessed through surveys of customers via telephone call, email or post.	Consume r
Response Time	The average time phone calls are answered; time it takes for a Help Desk agent who is to troubleshoot the service request to contact an authorized caller.	CRM, ACD
Call abandon rate	Percentage of calls where callers disconnect before reaching an agent	ACD
Employee Proficiency	Skill set of the Helpdesk analysts.	Agents
Call Volume	The number of calls taken by the Help Desk within a certain time period (a day, a month, a year).	ACD, CRM
Solution Accuracy	Assessment of the accuracy of solutions the Help Desk provides customers.	Agent, Consume r, CRM,
Reliability of Predefined Solutions	How reliable is the Knowledgebase data	Agent, CRM
Tracking Accuracy	Percentage of helpdesk cases resolved accurately	CRM
Resolution Time	Average Time it takes to resolve a problem	CRM
Resolution Excellence	The number of problems resolved versus the number of customer problems issued.	CRM

First Time Settlement	The number or percentage of problems resolved during the first customer call.	Agent, CRM, ACD
Number of calls	The number of calls taken per Help Desk agent per shift.	ACD
Time controller	The time spent per call.	ACD, CRM
Opened tickets	Number of helpdesk tickets opened per Helpdesk agent per shift.	CRM
Closed tickets	Number of helpdesk tickets closed per Helpdesk agent per shift.	CRM

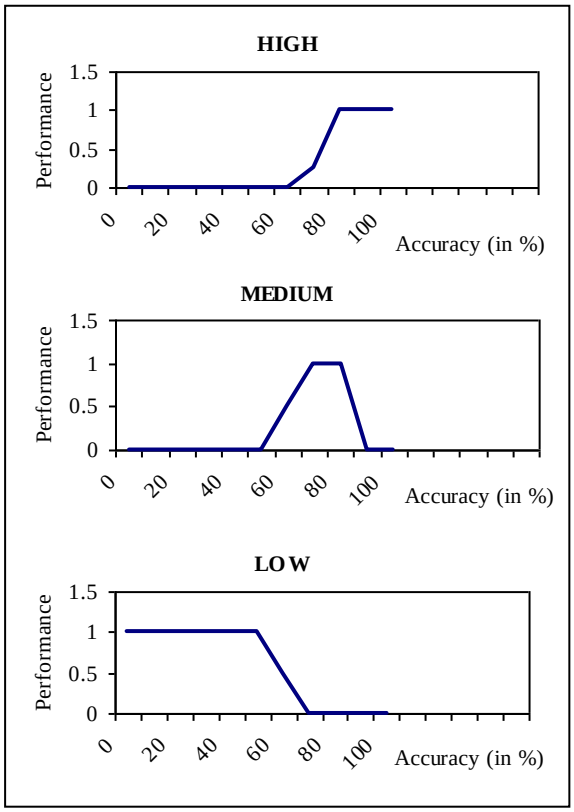
For the Helpdesk service, we will illustrate the fuzzification rules for two key metrics of the service, viz. the Resolution time and Tracking Solution Accuracy. For this service, the Response time fuzzification rules will be similar to the ones applied in the previous example, except that the time scale will be in minutes instead of seconds to account for the Human element of the service.



Graph 2: Fuzzification rules for Resolution Time

For our illustration, Resolution time performance is LOW if Resolution time is greater than 4 hours. It is MEDIUM if Resolution time lies between 2 hours and 4 hours. It is HIGH if Resolution time is less than 2 hours. We are using our experience in managing Helpdesks to set the HIGH, LOW and MEDIUM for this metric. Graph 2 illustrates the fuzzy rule for Resolution time performance metric.

Another important quality metric for Helpdesk service is 'Tracking Solution Accuracy' which measures the percentage of problems resolved accurately. This metric is completely dependent on Service agent's proficiency, or his/her skill set. The Service Agent could be dependent on (or coupled with) other services, like the Knowledgebase service or Expert Agent, for solutions and this dependency will also affect the Agent's performance which will have direct bearing on the Service Quality. For our illustration, 'Tracking Accuracy' is HIGH if 80% or more cases are closed accurately. It is MEDIUM if it lies between 65% and 80%. It is LOW if it is less than 65%. We are using our experience in managing Helpdesks to set the HIGH, LOW and MEDIUM for this metric, however this baseline can be adjusted based on the SLAs. Graph 3 illustrates the fuzzy rules for Tracking Solution Accuracy metric.



Graph 3: Fuzzification rules for Tracking Solution Accuracy

Similar Fuzzyification rules can be applied to the other performance measures listed in Table 2. Once we have

determined the fuzzy rules for our performance measures, we can create linguistic rules for the service that will help us determine the Service Quality. We are listing below some linguistic rules for the Helpdesk Service.

1. If {(CRM software performance is LOW) OR (Agent's performance is LOW) OR (ACD performance is LOW)} then the Helpdesk service performance is LOW
2. If {(CRM Software loosely coupled with Knowledgebase service) AND (Knowledgebase service performance is LOW)} then CRM Software performance is MEDIUM.
3. If {(CRM Software tightly coupled with Knowledgebase service) AND (Knowledgebase service performance is LOW)} then CRM Software performance is LOW
4. If {(Agent's proficiency is tightly coupled with Knowledgebase) AND (Knowledgebase service solution accuracy is LOW)} then Solution Accuracy is LOW
5. If {(Agent's proficiency loosely coupled with Knowledgebase) AND (Knowledgebase service solution accuracy is LOW)} then Solution Accuracy is MEDIUM
6. If {(CRM software is tightly coupled with Database service) AND (Database service performance is LOW)} then the CRM Software performance is LOW
7. If {(Database service is tightly coupled with Network service) AND (Network service performance is LOW)} then the Database service performance is LOW.
8. If {(ACD Software tightly coupled with Telecommunication service) AND (Telecommunication service performance is LOW)} then the ACD software performance is LOW
9. If ACD software performance is LOW then Response Time is LOW
10. If ACD software performance is LOW then Call Abandon Rate is HIGH
11. If CRM software performance is LOW then Resolution Time is HIGH
12. If {(Agent is loosely coupled with Expert's service) AND (Expert's Performance is LOW)} then Agent's Proficiency is MEDIUM
13. If {(Agent is tightly coupled with Expert's service) AND (Expert's performance is LOW)} then Agent's Proficiency is LOW
14. If {(Agent is tightly coupled with Expert's service) AND (Expert's Solution Accuracy is LOW)} then Solution Accuracy is LOW
15. If {Helpdesk Service performance is LOW} then Customer Satisfaction is LOW
16. If {(Response Time is LOW) OR (Call Abandon Rate is HIGH)} then Customer Satisfaction is LOW
17. If {Resolution Time is HIGH} then Customer Satisfaction is LOW
18. If {(Agent's Proficiency is LOW) OR (Solution Accuracy is LOW)} then Customer Satisfaction is LOW

Again, each of these performance rules should be evaluated for the service whose quality is being measured to determine if the Service is performing at its desired level or not.

3.4. Example for Infrastructure as a Service

Infrastructure as a Service (IaaS) is an instance that allows a user to request an appliance (hardware, Operating System, and applications or a subset of those) from a catalog of pre-defined solutions. Present incarnations of IaaS have varying levels of sophistication, for example, a simple implementation may allow a user to specify the size of the individual components (a Blade with 2 CPUs with 2 Cores/CPU, x GB RAM, y GB of attached storage, Linux, and DB2) while a more sophisticated implementation might allow the specification to be in terms of performance (something that will allow for a throughput of 100 transactions/minute with the ability to dynamically resize for 25% additional demand). Simpler implementation might also fulfill the request based on a first fit basis while more sophisticated implementations might optimize the provisioning to optimize resources or to implement look-ahead to fulfill anticipated requests.

Typically, IaaS involves all three service elements i.e. human agents, actual software that encodes the service, and other services/resources. For a typical IaaS implementation, there is a portal front end which allows the user to log a request. The request is routed for the necessary approvals and the request is fulfilled once the approvals are in place. IaaS typically relies on multiple supporting services and the coupling is far too extensive for the scope of this paper. We rather take a subset of the supporting services to elucidate the applicability of the proposed framework. The subset of the services, for example, includes agents who resolve user problems that are logged into a ticketing system, provisioning and virtualization software that encodes the service, and other services which are relied upon to deliver IaaS (such as Disaster Recovery, System and User Management Services, Account and User Management, Metering, License Management, and so on).

Table 3 lists some of the metrics that may be used to measure IaaS service quality.

The availability metrics in IaaS are the most complex which impact the provider as well as the consumer. This requires a careful planning of the infrastructure and accurate predictions of the distribution of the demand.

Table 3: Some performance metrics used by IaaS services

Metric	Measures what
Availability Metrics	Both in terms of uptime as well as the availability of a resource of a specific type (e.g. 128 core machine may not be available).
Utilization Metrics	Utilization factors of each resource type and whether utilization is skewed (large fraction of clients require compute resources leading to idling of network bandwidth)
Metering and Pricing Metrics	The pricing for the minimum committed resource usage and the pricing for dynamic sizing if available.
Latency Metrics	The time taken to fulfill the request or dynamically resize a request Catalog. The catalog defines the configurations that a user may request. A larger and richer catalog is typical of sophisticated IaaS installations.
First choice allocation rate	What percentage of the provisioning requests were accommodated
Mean time to resolution	Average Time it takes to resolve a problem
First fix rate	Percentage of problems that are fixed at first attempt

The utilization metrics reflect the quality of the original predictions of the distribution of demand though it can deteriorate due to the skewed nature of the on-boarded clients. When dynamic resizing is available to meet temporal peaks in the workload, how resource utilization is metered and priced, how long does it take to respond to the dynamic resizing requirement are all backend metrics that translate into front end metrics (for example, the richness of the available appliances, the rate of first allocations, etc.). The mean time to resolution and first fix rate are more traditional client end (front stage) metrics. It is obviously now possible to create linguistic rules that relate the metrics in Table 3 to service quality. Some examples of rules appear below.

1. If {(Availability Metrics are LOW) OR (Utilization metrics are skewed)} then Time to fulfill request is HIGH
2. If {Utilization metric skewness is HIGH } then First choice allocation rate is MEDIUM.
3. If {frequent need for dynamic sizing is HIGH} then Latency is HIGH and allocated resource usage is LOW.
4. If {frequent need for dynamic sizing is HIGH} the billing report complexity is HIGH the client's IT expenditure plan complexity is HIGH.

4. Ongoing Work

In this paper we presented a framework that can be used to relate metrics of the backstage in a service orchestration to the metrics at the frontstage. To the best of our knowledge, this is the first such effort, and it is critical since the front end is what the customer or the consumer sees, and on which SLAs and terms of the contract between the client and the service provider are typically negotiated. The framework is flexible, and allows instances to be created with rules elicited from domain experts. In ongoing work, we seek to validate this framework by applying it to not just transactional data, but also to elicit rules that capture business leaders' insights into how service accounts as a whole can provide quality.

5. Acknowledgement

The authors would like to thank C.H. Murthy, Nithya Rajamani, and Guruduth Banavar of IBM India Research Labs for the comments and feedback on the proposed approach.

6. References

[1] M Xu, Z Hu, W Long, W Liu, Service virtualization: Infrastructure and applications - The Grid: Blueprint for a New Computing Infrastructure By Ian Foster, Carl Kesselman, Morgan Kaufman, 2004

[2] LJM Coulthard, Measuring service quality: A review and critique of research using SERVQUAL, International Journal of Market Research, 2004

[3] A Parasuraman, VA Zeithaml, A Malhotra, ES-QUAL: a multiple-item scale for assessing electronic service quality, Journal of Service Research, 2005

[4] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous & Practical Approach. 2nd edn. Reading, 1997.

[5] N. Fenton and A. Melton, Deriving Structurally Based Software Measures, Journal of System Software, (12) 1990, pp. 177–187.

[6] H Dhama, Quantitative models of cohesion and coupling in software Journal of Systems and Software, Volume 29, Issue 1, April 1995, pp. 65-74

[7] G. Klir and B. Yuan, Fuzzy sets and fuzzy logic: theory and applications, Prentice Hall, 1995.

[8] P. Jain, R. Kothari, K. Ponnalagu, Association Rule Mining for Autonomic Solutions, Working paper.