Managing the Quality of Virtualized Services

Karuna P Joshi, Anupam Joshi, Yelena Yesha
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD, USA
{kjoshi1, joshi, yeyesha}@umbc.edu

***Abstract*—Managing the quality of virtualized services that are delivered on the cloud is very challenging. Such services are often composed of smaller components that are assembled on an as-needed basis. In this paper, we propose a framework to measure and semi-automatically track quality delivered by a Virtualized service delivery system. The framework 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 consumer interacts with the service. This allows administrators responsible for the functioning of a service to monitor its quality based on the measurements typically already done for the component services. The framework is general enough to be applied to any type of IT service. In the paper, we primarily concentrate on the Helpdesk service. We include the performance rules we have created by mining Helpdesk data.**

Keywords- Services Quality; framework; cloud computing

I. INTRODUCTION

The delivery of Information Technology (IT) services is moving away from a single provider model, and is increasingly based on the composition of multiple (other) services and assets. Often, the composed service consists of component services that come from specialized providers. In some cases, organizations utilize multiple service providers to mitigate risks associated with a single provider. In other cases, they may utilize a single provider who in turn utilizes the services of other providers. Moreover, the component service of a provider could simultaneously participate in several composed service orchestrations. The service, in effect, is virtualized and delivered from the service cloud. This virtualized model of service delivery [1] allows easier customization, better utilization, and greater responsiveness and is becoming the preferred method to deliver services ranging from the traditional helpdesk and back-office functions to services such as Infrastructure as a Service (IaaS). Indeed, the virtualized model of service delivery even extends to IT Enabled Services (ITeS) which typically include a large human element.

The virtualized nature of service delivery brings about new challenges in managing the quality of service delivery, which is a key responsibility of the administrators of the service/system. It is easier to express, understand, and manage Quality of Service (QoS) metrics related to individual services and the resources they use (such as storage, network, processing power, etc.). However, in the virtualized model, it is the overall and composed service that is experienced by the client or the business. It is the perceived quality at the front stage of this composed service that often acts as a differentiator in an otherwise commoditized environment. Moreover, Service Level Agreements (SLAs) related to customer satisfaction or other front end measures (response time, wait time, correctness, etc.) of the composed service are often used to manage delivery contracts and have revenue impacts for service providers. The administrators of the service thus need tools and techniques that allow them to track the quality of the overall service. This is hard to do, especially when the service composition could be done dynamically using orchestration languages such as BPEL and frameworks such as Microsoft's Service Factory [10] or IBM's WebSphere Portlet Factory [11]. While such systems are typically under the control of a single provider and tightly coupled today, the direction of evolution is towards composition of services from distributed, autonomous providers on the cloud.

This paper proposes a method for translating the metrics related to individual components of the service, like accuracy, responsiveness, uptime, etc. (which are in a sense backstage metrics) to the front stage experienced by the client or business. This is required to assess if the virtualized service will meet the SLAs, decide on what resources to allocate to it, and perhaps choose between providers when selecting component services to be composed. It will also allow a service provider to differentiate from competitors offering a similar service. It is generally easier to control the performance of an individual service which does not depend on any others. Its quality can be measured, and often correlated with the performance of resources upon which it depends. This has been the focus of much of the existing QoS work. However, the virtualized service delivery model requires the 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 control the performance of the overall service or provide quality measures for it in terms of the quality and performance of the underlying services.

Our proposed framework can be used by managers to get a holistic view of all the IT services being used in the organization including those which they do not manage or directly interface with. In a virtualized delivery environment, these services will be provided by various providers distributed across the globe. This comprehensive framework

will provide managers with a tool to manage these myriad of services and identify the component services causing bottlenecks in seamless flow of enterprise data. It will also help them identify the realistic thresholds that their IT service systems can operate under based on single component QoS measurements, as opposed to the ideal, almost utopian measures, which might have been set in the SLAs.

For this framework, we begin by identifying the front end performance metrics tracked by the SLAs of a service (or composite service). We next determine the dependency of this service (or composite service) on other services, or on backend applications and resources. We also identify the QoS measures tracked by these backend services. Linguistic rules are used to define how the front end measures of a service relate to its resources (or the other services it calls). Fuzzy Logic [7] is used to reason over such rules to move from the known component metrics to composed metrics (described in detail in section III). Such rules can either be elicited from experts, or be mined from historical data. When all the performance rules for a service have been linguistically defined, then the predicted quality can be dashboarded using standard applications and integrated into existing tooling to track services. Service administrators can then pro-actively track these rules by periodically polling the QoS measures of individual services and gathering the overall quality status of the service.

To demonstrate the capabilities of our framework, we apply it to the Helpdesk service to relate service quality to known backend measures. While a helpdesk today is outsourced to a single provider, it can be virtualized and provided by composing its component services (such as automatic call dispatch, human agents, knowledge base etc.) from separate providers on the cloud. We show performance rules that are created both based on domain knowledge and mining of historical data. We were able to access seven years of historic data of UMBC's IT Helpdesk system for our study. We mined Association Rules [6] from this data using WEKA [8] data mining tool and generated performance rules based on them. We also have summary data from a large financial service organization. We choose the WEKA tool to illustrate our framework as it's a freeware. One could also use other tools that generate association rules to identify performance rules

We have laid out the rest of the paper as follows. Section II covers related work in service quality. In Section III we detail our proposed framework and illustrate it in section IV by using example of a Helpdesk service. We conclude with an outline of our ongoing and future work.

II. RELATED WORK

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 or managing IT service quality in virtualized environments. Perhaps the seminal quality measure is SERVQUAL. It is a multi-item scale developed by Parasuraman et. al, [2] to assess customer perceptions of service quality in service and retail businesses. The scale is based on five dimensions of service quality, viz. Tangibility, Reliability, Responsiveness, Assurance and Empathy (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 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.

Other researchers have also proposed solutions for measuring quality for applications supporting e-business. Santos [12] proposes quality measures of service interface like ease of use, appearance, linkage, structure and layout, and content and other measures like reliability, efficiency, support, communication, security, and incentives.

However, these existing quality measures are not appropriate for IT/ITeS. For one, they are defined for Business-to-Client (B2C) 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 request 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 service metrics are largely confined to the front stage of a single service, both in terms of the objectively measured variables as well as the perceived measures such as quality. Given that a composed 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 tooling 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. They can also help determine where more resources might be needed. Second, they can help decide what new services can be provided from the given resources

III. PROPOSED FRAMEWORK

The framework that we propose for measuring the performance has three key parts – the elements of a service, the dependency between services, and rules that relate the performance of components. This general framework is applicable to every domain of IT services, and particulars of these parts can be instantiated for a given composite service.

To measure the quality of a service, we need to first identify the components of the service and the other services or resources that it is dependent upon. We next determine the coupling or dependency measure between the main service and the dependent services. Once we have identified the service and the coupling measures, we generate performance rules for the service that will enable us to capture the service quality.

A. Service Elements

Services comprise of three key elements, the agents or Human Beings involved in providing the service, the actual software that encodes the service provided, and other services/resources (software or hardware) 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.

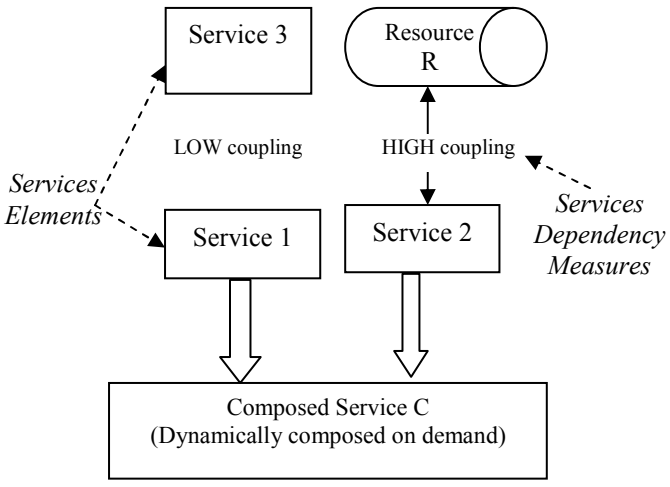
B. Dependency/Coupling Measures

The component services or agents in an orchestration depend on others in the orchestration for their performance. A component service also depends on the resources it uses. Thus a key component of the framework is to specify the dependency or coupling measure of a service on other services and components. Our aim is to 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 or deployed/used the service. Another option is to leverage work done in the software engineering domain on software coupling [4][5].

Service Coupling can be thought of as a measure that is similar to, but not the same as, the coupling measure used in traditional software engineering to describe the interdependence between two software modules. 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.

C. Performance Relation Rules

As briefly alluded to earlier, the performance of a service can be related by rules to the performance of other services it depends on and their degree of dependency, as well as the resources it needs. These rules are rarely crisp and normative – rather they are descriptive and uncertain. Much formalism has been developed for such uncertain rules, such as probabilistic logics, possibilistic logics, fuzzy logic, rough sets etc. We use fuzzy logic to encode these rules. Fuzzy sets provide us with the ability to classify elements into a continuous set using the concept of degree of membership. The characteristic function or membership function not only gives 0 or 1 for membership, but can also give values between 0 and 1. For instance, instead of expecting an exact numeric measure of dependence between two services, we could use a description like '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. A key issue here is to articulate these rules. This can be done by experts in the domain who understand the performance relations between components. In addition, such rules can often also be mined from the large volume of historical performance data. We have used both these approaches for our discovery of Helpdesk performance rules.



Performance rule: If {(Service 1 is loosely coupled with Service 3) AND (Service 3 performance is LOW)} then Service 1 performance is MEDIUM
If {(Service2 is highly coupled with Resource) AND (Resource performance is LOW) then Service 2 performance is LOW

Figure 1: Composed Services

Figure 1 illustrates the elements of our framework for a simple composition of a service C in a virtualized environment. Consider a Service C is that is composed of two mutually exclusive services Service 1 and Service 2. Service 1 is loosely dependent on Service 3 for its performance, i.e. Service 1 will continue to perform even if Service 3 fails. Service 2 is heavily dependent on Resource R and would fail if it's unavailable. Service C will most likely be created on demand to meet a client's requirements. We also list examples of what the performance rules will look like. While for this illustration we assume that Service 1 and 2 do not share any functionality or resources amongst each other, in some instances they could share resources and files. The logic of our framework will however stay the same if such a dependency exists since the only extra step needed would be to capture the coupling measure between service 1 and service 2.

IV. Applying the Framework to the Helpdesk Service

To illustrate our framework for the IT enabled services, we apply it to the Customer Relationship Management (CRM) or Helpdesk service. This service provides technical solution/assistance to its consumers. We use the helpdesk service as an example because unlike SaaS or IaaS, it involves both human and hardware/software agents. For examples from SaaS and IaaS domains refer to [13]. A Helpdesk is also the kind of service most likely to be delivered remotely. While at present it is from a single provider, in future such services will come from the cloud, with potentially different providers providing separate components like ACD, human agents, and the knowledge base. Recall that the objective is to find rules that would let us predict the quality of a composed service based on QoS measures of individual components.

To this end, we were able to access our University's IT Helpdesk's historical data of around seven years. The Helpdesk was managed using BMC's Remedy software. The service was provided by twelve support groups and generated around 20,000 tickets per year. As expected, the call volume was high during the beginning of the semester. It was also high around the rollout of any major software and the Helpdesk data was tagged to reflect that.

Unlike many larger helpdesk services, UMBC did not have a classic Automatic Call Distribution (ACD) system, since most of the tickets are generated via email. However, the component that assigns tickets to particular agents (using a mix of humans and automated techniques) behaved very much like an ACD.

The UMBC Helpdesk did not have a knowledgebase integrated with their Remedy system, so the agents could not refer to a predefined list of solutions. The agents however were highly knowledgeable in specific domains, and also had access to external experts via software support agreements.

We were also able to access historical Helpdesk data of a large international financial organization that used Remedy as well to manage their CRM system. This IT Helpdesk had implemented the ACD system since most of their consumers contacted them via phone. They had also integrated a knowledgebase package with their Helpdesk system which they utilized to auto-populate the solution field.

To obtain the rules relating the quality of this service to its component elements, we followed these steps.

A. *Identified the Service Elements*

After reviewing with domain experts, we identified the key elements of Helpdesk service which are illustrated in Figure 2. These include agents that provide the actual service of responding to the consumers, software used to provide the service which is a CRM application, and Automatic Call Distribution (ACD) software integrated with PBX (Public/private Branch Exchange or Telephone switching) used to automatically route the Helpdesk calls to the various agents. These elements were identified based on service capabilities.

B. *Determined the Dependencies/Coupling*

We deduced the CRM software dependencies (see Figure 2) from its configuration and resource needs, like database, Operating System or Network requirements. Agent dependencies were gathered from domain experts. The CRM software depends on a DBMS application to manage the data and cannot function without it. The DBMS application in turn relies on the Network applications to manage the communication with user sessions, other applications and servers. In the market today there exist many software packages that provide predefined solutions or a knowledgebase for the most common IT problems that a user faces. Alternatively, many organizations also build in-house versions of knowledgebases that can be used to automatically populate solution fields in the CRM application. It has also been observed that a Helpdesk agent performs better if s/he is able to access a subject expert to confer about the solution if the problem is complex. So we determined that the Helpdesk service is coupled with the following external services.

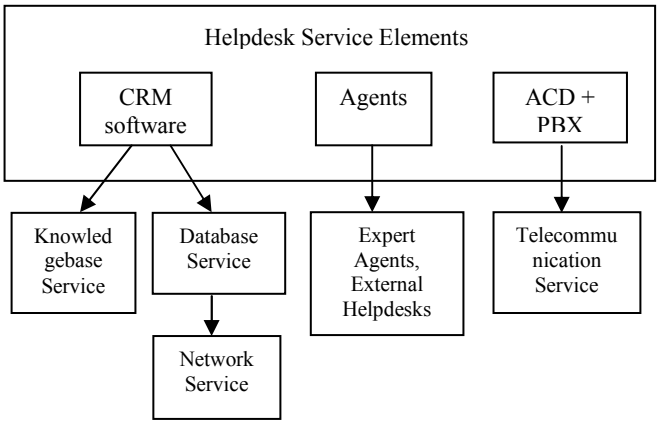


Figure 2: ITeS example: Helpdesk service

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, as well as the subject matter knowledge of the agents. The Knowledgebase could be a set of pre-determined solution list or FAQs or Help systems. Depending on the implementation it can be a HIGH or LOW coupling with the CRM application.

e. ACD+PBX software is TIGHTLY coupled with the underlying Telecommunication service.

C. Determined the Helpdesk Metrics

To identify the key performance metrics that are tracked for the Helpdesk service, we referred to domain experts and industry standards [9]. Table 1 lists the primary performance metrics used in the Helpdesk services industry today to measure the service quality. Depending upon the configuration of their Helpdesk, organizations track some or all the metrics listed in the table.

TABLE I. PERFORMANCE METRICS USED BY HELPDESK SERVICE

Metric	Measures what	Helpdesk Element
Customer Satisfaction	Assessed through surveys of customers via telephone call, email or post.	Consumer
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, CRM, Consumer
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

D. Developed the Performance Rules

Our next step was to develop the performance rules to measure the service quality perceived by the customer as a function of the quality measures of the component elements of the helpdesk service. For most services, domain experts have rules of thumbs about factors that affect the perceived service quality. For the Helpdesk service, some such rules given by experts are:

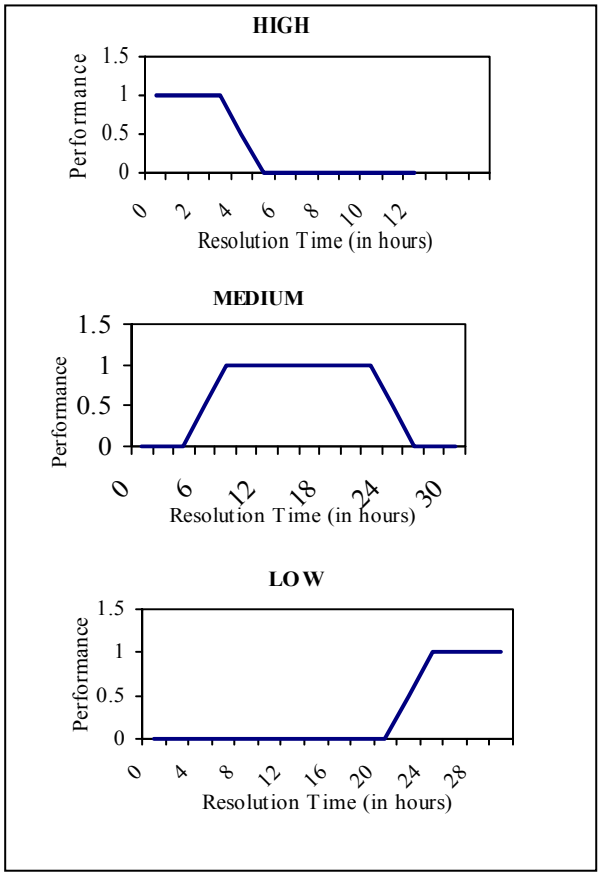
1. If {Helpdesk Service performance is LOW} then Customer Satisfaction is LOW
2. If {(CRM software performance is LOW) OR (Agent's performance is LOW) OR (ACD performance is LOW)} then the Helpdesk service performance is LOW
3. If {(CRM software is tightly coupled with Database service) AND (Database service response time is LOW)} then the CRM Software response time is LOW
4. If {(Database service is tightly coupled with Network service) AND (Network service performance is LOW)} then the Database service performance is LOW.
5. If {(ACD Software tightly coupled with Telecommunication service) AND (Telecommunication service call drop rate is HIGH)} then the ACD software performance is LOW
6. If ACD software performance is LOW then Response Time is LOW
7. If ACD software performance is LOW then Call Abandon Rate is HIGH

We argue that such performance rules can also be automatically generated by mining historical performance data where it is available. In particular, for helpdesks, we can mine rules that associate customer satisfaction with the measured quality of the helpdesk components. For this effort we used the WEKA [8] tool to generate the association rules from the historic Remedy data mentioned above. Initially we included every field of the Helpdesk record (called ticket) for the study. In the next iteration, we removed the fields that were either redundant (duplicate fields tracking the same information) or too large (like the comments field which had free text) for the tool to handle. We also created three new fields called response time, resolution time and time taken. The response time was calculated as the difference between the ticket assignment time and ticket creation time. The time taken was calculated as the difference between the ticket resolution time and ticket assignment time. The 'resolution time' field was then set as HIGH, MEDIUM or LOW depending on the calculated resolution time. These fields are providing measurements of how quickly a problem was addressed, which is a proxy for the perceived quality of the service.

We next ran the associate function in WEKA for multiple datasets spanning seven years from 2000-2007. The important fields that generated association rules include Ticket Priority, Ticket Origin, Customer Position, Resolution

Time, Response Time, Submitter, Assigned to, Group and Department.

Since the UMBC helpdesk did not maintain explicit customer satisfaction feedback linked with tickets, we used a low resolution time and high solution accuracy as proxies for the service quality. So we looked for factors that influenced these two measured values. We note that a majority of the consumers requested the service using email which was closely integrated with the Helpdesk application. The mail system was set up to automatically assign the ticket to the appropriate agent. For instance, issues related to audio-video services were automatically routed to the agent dedicated to that task. Association rules generated showed that this feature enabled reducing the resolution time. For majority of the tickets, the agent who was assigned the initial ticket was also the one who resolved the ticket. This also helped keep the resolution time low. This is likely because the initial assignment was to the appropriate agent, who in turn had the knowledge/resources to correctly resolve the problem. So clearly, the accuracy of the ticket assignment/ACD component, and the knowledge of the agent are both important for the overall customer satisfaction.



Graph 1: Fuzzification rules for Resolution Time

Some association rules that were generated are listed below.

- Ticket Was Assigned to → Ticket Assigned to confidence:(0.98) (The ticket mostly stayed assigned to one analyst.)
- Ticket Assigned to → Ticket closed by confidence:(0.98) (The ticket was mostly closed by the person that the ticket was assigned to or by a automatic trigger.)
- Ticket Assigned by Mailer demon → Priority=Normal confidence:(0.98) (Majority of tickets were automatically assigned and were set to Normal priority.)
- Ticket Origin=Email Initial Assigned To=HelpDesk:Mail Assigned by=Mailer daemon → Priority=Normal confidence : (0.98)
- Cust Position=Staff resolution time=HIGH → Priority=Normal confidence:(0.96)

We also got rules indicating that some of the groups (areas of agent competence) were associated with low resolution times, whereas others were not (the exact rules are not given to protect privacy).

We reformulate the association rules we got from this analysis as fuzzy linguistic rules to make them more general. Recall that while we are mining some historical performance data of an instance of the helpdesk service, we want rules that can be used to predict the performance of some future helpdesk service that is composed from similar component elements. Hence the need to take the very specific rules that model the historic performance of the instance and create more general rules that can be used later. For instance, we can fuzzify the "resolution time" performance metric into the fuzzy variables HIGH, LOW and MEDIUM based on the rough service level guidelines followed by the University's Helpdesk. Graph 1 illustrates the fuzzification function for the resolution time performance metric. The resolution time is regarded excellent if it is less than 4 hours. In such a case, the performance of the metric is HIGH. Performance is MEDIUM if resolution time lies between 4 hours and 24 hours (or one day). It is LOW if resolution time is greater than one day. There were no formal SLAs for ticket resolution for the UMBC helpdesk. Where SLAs are available, they can provide guidance to the fuzzification functions.

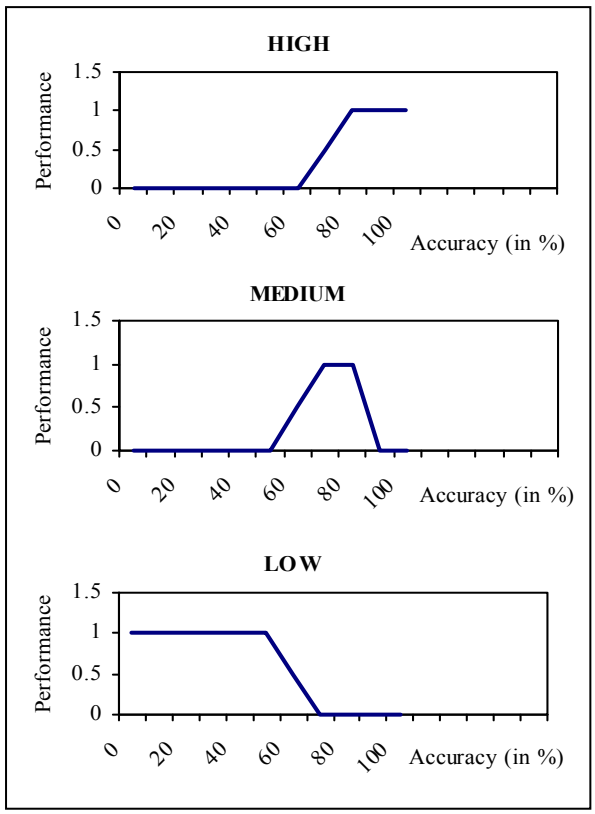
At present, we have manually fuzzified the association rules. This required us to be aware of the Service level guidelines/agreements employed by the service provider (in our instance the University's helpdesk) and the knowledge of the terms specific to the service domain (in our example Helpdesk service). However, there are a variety of automatic fuzzification techniques in literature, for instance the fuzzy c-means based technique proposed by Sugeno and Yasukawa [14].

Some performance rules that can be elicited from these association rules are

1. If {Resolution Time is LOW} then Helpdesk Service performance is HIGH
2. If {(Agent assigned ticket = Agent closing ticket) AND (Agent proficiency is HIGH)} then Resolution time is LOW

3. If {ACD performance is HIGH} then Service performance is HIGH
4. If {Assigned group is (X)} then Service performance is HIGH

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 and the complexity of the problems coming in. 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. We referred to the historical data from the financial organization to determine the fuzzy rules for this metric. For our illustration, we regard that 'Tracking Accuracy' is HIGH if 80% or more cases are closed accurately. It is MEDIUM if it lies between 65% and 80%. It



Graph 2: Fuzzification rules for Tracking Solution Accuracy

is LOW if it is less than 65%. Graph 2 illustrates the fuzzification function for Tracking Solution Accuracy metric.

Some performance rules that can be elicited from this metric include.

1. If {(CRM Software loosely coupled with Knowledgebase service) AND (Knowledgebase service response time is LOW)} then CRM Software response time is MEDIUM.
2. If {(CRM Software tightly coupled with Knowledgebase service) AND (Knowledgebase service response time is LOW)} then CRM Software response time is LOW
3. If {(Agent's proficiency is tightly coupled with Knowledgebase) AND (Knowledgebase service solution accuracy is LOW)} then Solution Accuracy is LOW
4. If {(Agent's proficiency loosely coupled with Knowledgebase) AND (Knowledgebase service solution accuracy is LOW)} then Solution Accuracy is MEDIUM
5. If CRM software performance is LOW then Resolution Time is HIGH
6. If {(Agent is loosely coupled wih Expert's service) AND (Expert's Performance is LOW)} then Agent's Proficiency is MEDIUM
7. If {(Agent is tightly coupled with Expert's service) AND (Expert's performance is LOW)} then Agent's Proficiency is LOW
8. If {(Agent is tightly coupled with Expert's service) AND (Expert's Solution Accuracy is LOW)} then Solution Accuracy is LOW
9. If {(Agent's Proficiency is LOW) OR (Solution Accuracy is LOW)} then Customer Satisfaction is LOW

Another key performance metric that we also studied is the "response time" metric. We fuzzify it into the fuzzy variables HIGH, LOW and MEDIUM. For example, Response time performance is LOW if response time is greater than 15 minutes. It is MEDIUM if response time lies between 5 and 15 minutes. It is HIGH if response time is less than 5 minutes. The advantage of this metric is that it can be applied across the main helpdesk service as well as the services it depends on. A key point to be noted is that while the baseline for the various components of the services will be different, it is the SLA of the composite service that will be monitored by the consumer organization. Thus while a response time of 2 seconds might be regarded as a LOW performance for underlying Network component of the Service based on the Quality of Service (QoS) measures, it will still point to HIGH overall performance for the helpdesk service and the SLA of the composite service will be met.

Such rules can be used to predict the performance of a new helpdesk service. If we know that the ACD component is good, for instance, the service will be good. In fact, by using a fuzzy reasoner we can provide qualitative insights to the service manager even before a service is deployed. We can also help the service manager pick specific components where more than one are available that satisfy the orchestration specified. More importantly, it points out room for improvement. Clearly, tickets assigned to some service groups were not leading to quick resolutions in UMBC. This shows where further training and investment is needed. Further, it shows what SLAs are appropriate for a new service designed using the existing components. So for instance, if the high performing groups are used in a new service provisioning, higher SLAs can be promised compared to if the not so well performing groups are used. The financial organization helpdesk had a set the SLA for their response time to 30 seconds. They met their target on

average 73% of the times. They recognized that improving this metric would improve their overall performance.

Another avenue that can be explored is co-relating service performance rules with the customer satisfaction (CSAT) results. We did not have any CSAT data pertaining to the incident/ticket level for us to be able to illustrate this in this paper. However, if service providers are also tracking some of the service metrics via CSAT surveys then they can co-relate the performance rules with the CSAT levels.

V. Conclusion and 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 front-stage. 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. As such, a framework which can predict the quality of a composite service delivered from the cloud to an end user is of great importance to administrators and service providers. The results of such a technique can also be integrated into tools that are used by the administration community to monitor single services.

The framework is flexible, and allows instances to be created with rules elicited from domain experts or mined from historical performance data. We show the capability of the framework by describing instantiations of the framework for Helpdesk service. In ongoing work, we are continuing to validate this framework against the historical Helpdesk data that we have access to. In particular, we hope to shortly have access to detailed data from the international financial organization and mine that for performance rules. We are also building a fuzzy reasoner with Jess to automate the predictive part of this framework.

Acknowledgment

The authors would like to thank Jack Seuss, and Mike Carlin of the Division of Information Technology, UMBC, for their support in providing us data for our research. This work was supported in part by the Air Force Office of Scientific Research under MURI award FA9550-08-1-0265

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of 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 edition Reading, 1997.

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

[6] Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules, Proceedings of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, Sept. 1994.

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

[8] Weka 3 data mining software, University of Waikato, http://www.cs.waikato.ac.nz/ml/weka/, accessed on June 10 2010.

[9] Help Desk Balanced Scorecard Metrics by AKS-Labs, http://www.strategy2act.com/solutions/helpdesk_scorecard_excel.htm , accessed on June 10 2010

[10] Microsoft Web Service Software Factory, http://msdn.microsoft.com/en-us/library/bb931187.aspx, accessed on June 10 2010.

[11] IBM Websphere portlet factory, http://www-01.ibm.com/software/genservers/portletfactory/, accessed on June 10 2010.

[12] J. Santos, "E-service quality: a model of virtual service quality dimensions", Managing Service Quality, Vol. 13 Iss: 3, pp.233 – 246, 2003

[13] K. Joshi, A. Joshi, Y. Yesha, R. Kothari, A Framework for Relating Frontstage and Backstage Quality in Virtualized Services, UMBC Tech Report TR-CS-09-01, May 2009 .

[14] M. Sugeno and T. Yasukawa, A Fuzzy-Logic-Based Approach to Qualitative Modeling, IEEE Transactions on Fuzzy Systems, 1:1, pp 7-31, Feb 1993.