The Relationship Between Quality of Service and Business Metrics: Monitoring, Notification and Optimization

Katinka Wolter and Aad van Moorsel
TU Berlin, Sekr. FR 2-2
Franklinstr. 28/29,
10587 Berlin, Germany

Hewlett-Packard Laboratories,
1501 Page Mill Rd., MS 1U-14,
Palo Alto, CA 94304, USA

January 5, 2001

Abstract

This report discusses the effects of quality of service degradations on the profitability of e-services. We characterize possible relationships between quality-of-service metrics (throughput, delay, availability) and business metrics (revenue, costs). We build an e-service (named Q2B) that collects run-time data and visualizes the above relationships. Furthermore, the Q2B service triggers alarms if customers are likely to react to quality-of-service perturbations of a service or its suppliers. We also apply mathematical programming to facilitate business decision making based on the impact of QoS on business metrics.

1 Introduction

In the era of e-services, down time of a web server means that hosted e-commerce sites loose business. Excessive delays turn into dissatisfied customers that loose patience, and will not buy on a site (the infamous eight or ten seconds deadline [BBK00a, BBK00b]). In other words, QoS becomes more and more intimately related to the bottom line of a business. We denote this relationship as the 'Q2B' relationship ("quality of service to business"). For IT and business managers there is a growing need to track the Q2B relationship at run-time, to obtain insight in the consequences of

QoS alterations on the bottom line. From a management perspective, this implies monitoring both QoS and revenue or cost, and identifying the statistical correlation between the two. Based on such information, one may be able to tune a system or business process judiciously, thus bridging the gap between IT management and business management.

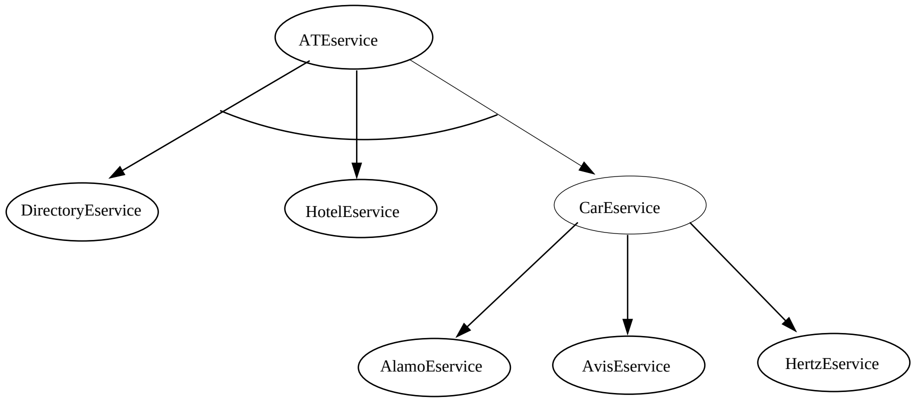

Figure 1: Accidental Tourist services configuration

We describe what we are after for the example of the 'accidental tourist' e-service [MDGH99]. The accidental tourist service *ATEservice*, is an electronic travel service that has the structure depicted in Figure 1. In this figure, each ellipse denotes a service, and each arc indicates a transaction between services. The circle section connecting the outgoing arcs of ATEservice denotes that an AND operation takes place: the accidental tourist service issues transactions to all its children. The child-services are an e-service to make hotel bookings, an e-service to rent a car and the directory service, that coordinates incoming proposals and decides whether to accept them or not. The CarEservice is in fact a car broker that interacts with three competing car companies. Between those three services there exists an 'OR' relationship: the CarEservice will use only one of the three competing car rental e-services.

Some obvious business considerations take place in this system: end customers will use the ATEservice more if it provides good deals; Alamo competes with Hertz and Avis for deals with the CarEservice, etc. More hidden is the topic of our discussion: the economic consequences of poor performance or reliability. For instance, CarEservice may loose customers if the suppliers (Hertz, Alamo, Avis) perform poorly. In turn, ATEservice may also loose its customers, because its performance depends on that of its suppliers (among which is the CarEservice). We thus see that rather complex dynamics between a service, its suppliers and its customers determine how QoS relates with the financial bottom line.

Many different tools and approaches are known for monitoring QoS in e-commerce [PJA$^+$00, BC99, Obs, Fir], but existing work in system management rarely considers the business impact of QoS. If business aspects are considered it is often based on service level agreements (SLAs) [SLA, Hil93]. SLAs, however, are predominantly *static* in nature; that is, it is predefined how customers pay at various QoS levels. Customers can not influence this by changing suppliers. In this paper, on the other hand, we are interested in *dynamic* relationships between QoS and business metrics: customers respond dynamically to QoS changes by changing suppliers, thus impacting the business results of e-services. Also closely related, but different in nature, is some work on pricing of services: resource prices change if congestion is high [MB00]. Pricing is an interesting tool to influence customer decisions; we will not discuss pricing in this paper, but it certainly deserves further attention within the context of federated e-services.

Under the assumption that the relationship between QoS and business metrics is important and existing, we try to answer the following questions in this report: (1) can we instrument a system of federated e-services, monitor it, and visualize the Q2B relationship, (2) can we notify a business manager if QoS starts influencing the bottom line, and (3) can we adapt the business process to improve our overall gain in case of QoS changes? The following three sections provide the (positive) answers to those questions.

2 The Q2B Management E-Service

To obtain the data necessary to correlate QoS with business metrics, we instrument e-services that are implemented in e-speak middleware [esp]. This is part of an existing management system, and in Section 2.1 we describe how the Q2B service fits in this management system. To include business metrics, we extend the XML documents of the management system, as discussed in Section 2.2. Section 2.3 then illustrates the visualization possibilities offered by the Q2B e-service.

2.1 E-Service Management System

The management system as introduced in [MDGW00] follows the structure shown in Figure 2. It consists of several independent processes, possibly running on different machines. These processes communicate by exchanging XML messages over HTTP using e-speak [esp] and Chai server [cha].

As depicted in Figure 2, we use a simulation of the *Accidental Tourist* travel agent e-service to obtain experimental results (see Section 1). In this simulation, transactions follow a random path through a hard-coded service and transaction model. The response times of the single e-services are simulated as random delays between consecutive messages. The response times are, however, monitored through instrumentation. Furthermore, interceptors that are plugged in the e-speak middleware set timestamps and send these to the management e-service as well. The management e-service re-

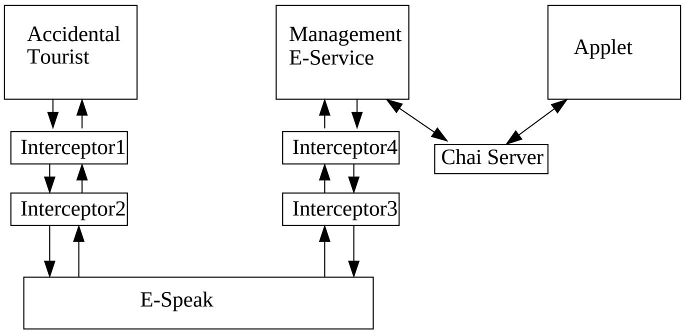

Figure 2: Structure of the management system

ceives on one hand data from the accidental tourist simulation and on the other hand requests from different user applets (see the right-hand side of Figure 2). Depending on the specifics of the requests, data from the accidental tourist service is passed to some of the managers in the management e-service. These managers then process the data and reply to the applets with the appropriate data. In the management system prototype, there already exists a performance manager, to which we added the Q2B manager.

To close this section, Figure 3 introduces in some more detail the control flow of the accidental tourist simulation. The first path of transactions is created by running *call-for-proposal* transactions on all but the directory e-service, which does some administrative jobs. Then the proposal is accepted with a certain probability (the arc labelled *accept*) and refused otherwise. If

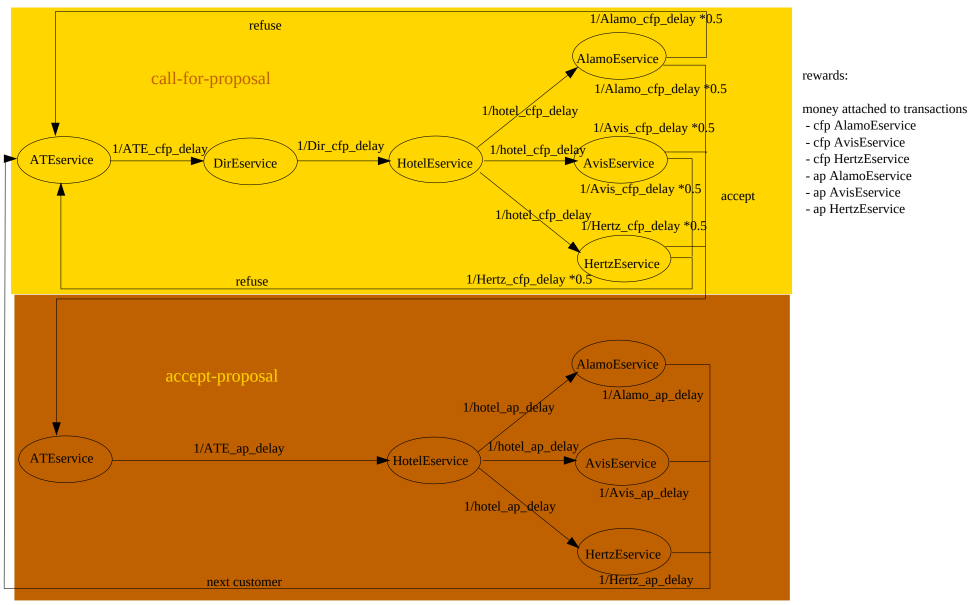

Figure 3: Flow Chart of the Accidental Tourist

the proposal is accepted, the second set of transactions, the *accept-proposal* transactions are issued. In the graph we close the loop which means that a new customer can enter the system.

2.2 Adding Business Metrics to E-Service Management XML

The e-service management system receives data using XML messages. These messages it either receives from the services to be managed or from the underlying middleware directly. In this section we show how to include business metrics in the XML management messages. Before doing so, we explain in detail what role XML plays in the management system.

The content of the XML documents depends on the parties involved in the communication, and the phase they are in. Figure 4 represents the different parties and communication phases. For managed and management service, it also gives the main elements of the corresponding XML documents.

The e-commerce system registers with the management e-service and publishes its *service model* and *transaction model*. The service and transaction models have been defined in the e-service management vocabulary of

[MDGW00]. Informally, a set of services exists, with transactions defined between services. With each service and transaction, attributes are associated, so that all necessary information can be derived for a service or transaction. In addition to the service and transaction model, the service *features* that will be monitored are specified. Features are characteristics of transactions, such as average response time or transaction costs. Note that registration is depicted with a dashed line because it happens only once, at initialization.

At run time, the e-commerce system continuously sends XML messages with *system information* to the management e-service. Part of the system information are the *metrics*. Metrics are the values of the transaction features; examples are '20 milliseconds' for response time, or a dollar amount for the transaction costs.

The management e-service then is queried for certain *features* by a user. These are the XML *requests* being sent to the management e-service. The management e-service responds with a *reply* message containing the *metrics*, that the management e-service computed from the system information.

Figure 4: XML contents and message flow

When adding business metrics to the management system, our main assumption is that revenue or loss can be associated with e-service transactions. (In cases where a transaction has nothing to do with money this value will be zero). The value of the transaction may be set during the transaction (the shopping basket scenario), or may be specified in a contract document (per hour of use, per unit of bandwidth, per use (newspaper), per month (subscription)). We note that for our research, the latter case is only of interest to us if customers have the opportunity to alter how much they pay by changing their chosen suppliers.

Since we associate business metrics with transactions, we create a new **Feature**, of type `Monetary`, as follows:

Feature

`Type:  Monetary`

```
Name:  AccumulatedRevenue
       AccumulatedLoss
       AverageRevenue
       AverageLoss
```

The `Monetary` feature is used in two places: at registration to denote what data can be provided, and at user requests to denote for what elements results are desired. The latter case leads to the following sample XML request:

```
<ServiceManagementFeatures Service = "Accidental Tourist">
  <TransactionManagementFeatures Transaction = "accept-proposal-hotel">
    <Feature Name = "AverageResponseTime"
       Type = "Performance"
       Duration = "20000">
    </Feature>
    <Feature Name = "AverageRevenue"
       Type = "Monetary"
       Duration = "20000">
    </Feature>
  </TransactionManagementFeatures>
</ServiceManagementFeatures>
```

The above XML asks for the e-service `Accidental Tourist` and the transaction `accept-proposal-hotel`. Over a sliding time window of 2000 milliseconds (`Duration`) both the feature `AverageResponseTime` of type `Performance` and the feature `AverageRevenue` of type `Monetary` are requested.

Corresponding to the **Feature** there is a **Metric** for the transaction instances:

Metric

```
Type:  Monetary
Name:  AccumulatedRevenue
       AccumulatedLoss
       AverageRevenue
       AverageLoss
```

Features and Metrics thus have the same vocabulary elements, but only Metrics have values. This is used in two ways: when an e-service sends results to the management service, and when the management service sends results to the user (labelled *Reply* in figure 4). We show an example of both cases. In the first example, the management e-service provides the user with results for performance as well as business metrics. We note that, besides the monetary value two other attributes are associated with a transaction instance. One is the execution time of a transaction instance and the other is the success code, whether the execution could be finished successfully or not. In the example, the values for the requested metrics are *AverageResponseTime* equals 1000 and *AverageRevenue* equals 450.0.

```
<ServiceState Service = "Accidental Tourist">
  <TransactionState Transaction = "accept-proposal-hotel">
    <Metric Name = "AverageResponseTime"
       Type = "Performance"
       Duration = "20000">
       1000
    </Metric>
    <Metric Name = "AverageRevenue"
       Type = "Monetary"
       Duration = "20000">
       450.0
    </Metric>
  </TransactionState>
</ServiceState>
```

The final example in this section shows the use of a `Metric` through an XML message sent by a managed e-service to the management e-service. Note that we only show part of a much more extensive message that include sender and receiver information and other 'overhead'.

```
<ServiceState Service = "Car">
  <Service Instance = "Avis">
    <TransactionState Transaction = "accept-proposal-hotel">
      <TransactionInstance Id = "1234">
        MonetaryValue = 75
      </TransactionInstance>
    </TransactionState>
  </ServiceInstance>
</ServiceState>
```

In figure 4 we call this kind of messages *System information*. The above XML excerpt includes one new term, namely *MonetaryValue*. An e-service cannot compute the actual *Metrics* (which typically are averages or totals), but provides information on a per-transaction basis. In the example, the *MonetaryValue* of 75 is for an 'accept-proposal-hotel' transaction carried out on service 'Avis,' with '1234' as 'Id'. The management e-service can then compute the overall result for the registered metrics, and sends the desired metrics as reply after a user request.

2.3 Visualization

The user can view the information from the management e-service through a web browser running applets. In this section we present some examples of the visualization aspects of the management system. We have extended the applets of the existing management system, so that they show business metrics in addition to performance metrics. Thus we obtain applets that show QoS and business metric together.

The Kiviot plots (using terminology from Kant [Kan93]–they are also known as 'radar' plots) in this section show a set of *Metrics* that are averages over a sliding time window of predefined size (as specified by the XML request, see Section 2.2). If the system is operating well, we expect to see at all times a plot of star-like shape as shown in figure 5. If one of the services is

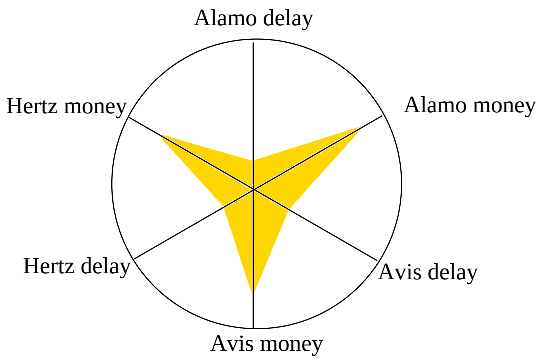

Figure 5: Desired shape of the Kiviot plot

not operating well its delay increases, as it does for *Avis* in figure 6. Then simultaneously the revenue (denoted as *Avis money*) decreases.

The data used for these plots is taken from the simulation of the accidental tourist. In the simulation, we artificially decrease performance of one service,

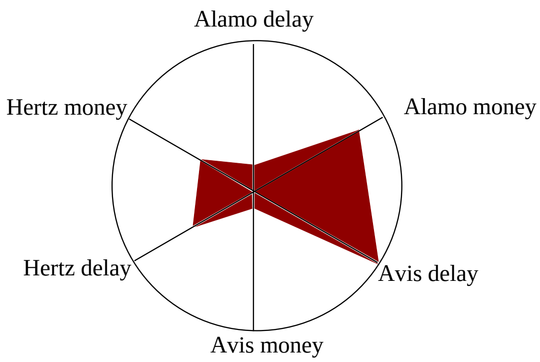

Figure 6: Observed Kiviot plot

and then assume that customers will choose a competing service instead. Note that one would like to be able to demonstrate that this is according to the operation of real e-commerce systems. We refer to [] for illustrations of how users respond to performance perturbations. Here we only want to demonstrate that we can monitor and visualize such effects. In particular, we show that the shape of a kiviot plot gives an intuitively understandable picture of whether a system is operating well or not.

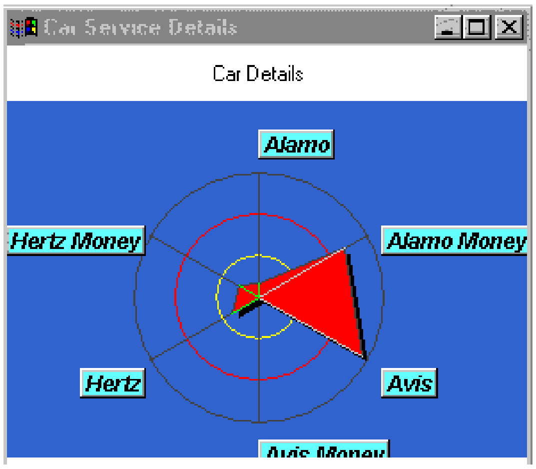

Figure 7: Screen shot of the Kiviot plot

A screen shot of our implementation as Avis' performance is turning bad is shown in figure 7. In the simulation, the first choice of a rental car company (out of the three options) is made randomly. However, if the sampled delay exceeds some threshold, the transaction is not completed and instead a different car company is chosen.

The delay is monitored and transmitted to the management e-service, but no money is made through this transaction. It would in this place be desirable to have a way of choosing the optimal service on which to run a transaction. Unlike the simulation that processes only a single job at a time, a real system has a load that should be distributed over a choice of services in an optimal way. We will address this issue in the next section.

The counters in Figure 8 display the totals of the metrics over the time the service has been operating. These numbers are based on collecting data for certain transactions (see figure 3). The number of hits is for instance determined as the number of *call-for-proposal* transactions that are run on the ATEservice. The number of orders placed is the number of *accept-proposal* transactions run on that same e-service.

We associate costs with the call-for-proposal transactions and revenue with the accept-proposal transactions. Accumulating over each gives the values for the *costs* counter and the *revenue* counter, respectively. The loss is determined as the monetary value attached to accept-proposal transactions that were unsuccessful.

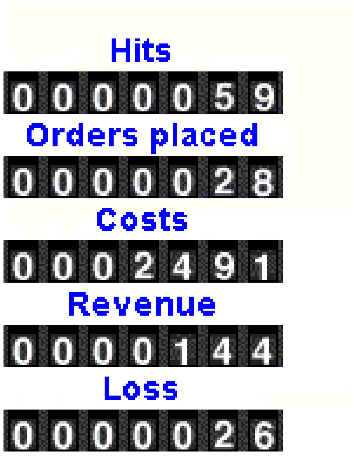

Figure 8: Screen shot of the counters

Plots such as the above help a human operator or business manager to quickly and easily get a picture of the 'financial health' of the system. This is a first step to good management of the e-commerce system. However, influencing the e-commerce system automatically based on the shape of a Kiviot plot or numbers in a counter is not straightforward, if not impossible. Therefore, we discuss in the next section the statistical correlation between delay data and monetary data. We use this to give a warning signal when poor performance of a system causes loss of revenue.

3 QoS-Based Alarms for Business Degradation

The first step in generating alarms is to identify the metrics depending on which action has to be taken. Then these metrics have to be monitored and reasonable thresholds for alarms have to be set up. We assume that business shrinks if performance of the system decreases, and, hence, we trigger alarms based on QoS metrics. The ultimate goal, however, is to improve business results. The way we address this issue is to first determine the correlation of performance and business metrics, determine the thresholds in business metrics which are assumed to be critical, translate them into performance thresholds according to the correlation function and then monitor performance and give alarms when thresholds are crossed.

Estimation of the correlation between performance and monetary value can be done in different ways. One can use linear or non-linear regression analysis to determine either a constant, linear, or polynomial relation between pairs $(p_i, \$_i)$ of performance values and monetary values.

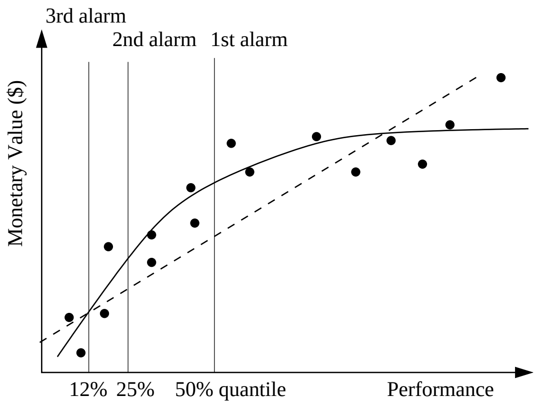

Figure 9: Estimation of performance and monetary value correlation

Figure 9 shows a possible scenario. The dots are observed pairs of performance and monetary value data. In this case, a high value on the horizontal axis indicates good performance, and a high value on the vertical axis indicates high revenue. We note that if no individual transactions can be observed, the observations can be replaced by their mean values. Each dot in Figure 9 then stands for the mean performance and the mean monetary value over a certain time interval. In any case, regression analysis can be performed like in Figure 9. In this figure, the dashed line gives a linear regression curve, whereas the solid line represents a non-linear, polynomial regression curve. The correlation factor in this situation would be close to one.

There is a variety of ways to determine when to trigger alarms using the statistical correlation between the QoS and business metric. Alarms based on the value of the business metric may be used to inidicate that one should think of improving the QoS of the service. The other way around, triggers based on the QoS values may be used to indicate (and even predict) business problems. The latter is only useful if the values of the QoS metric allow quicker detection of business problems than the business metrics themselves. One situation in which this may be the case is in the shopping cart scenario. Then, one can imagine that poor performance almost certainly implies that some customers will stop ordering; alarms tiggered based on the performance metric may be quicker in identifying this business problem. Obviously, such scenarios depend on the specifics of the e-commerce system and the statistical characteristics of the observed metrics.

Figure 9 illustrates the procedure of triggering alarms. First, one determines a threshold monetary value, for instance based on earlier business results or on some predefined (fixed) business target. In the figure tis is marked by monetary value $\$_x$. Then one finds the corresponding QoS value using the correlation curve. This result in value p_x. Now, if samples smaller than p_x occur, an alarm will be sent. Again, details of this procedure can be fine tuned depending on the particular situation.

4 Optimizing E-Service Business Operation

As mentioned in the previous section an e-service aims at optimal customer satisfaction in order to maximize its own profit. We assume that customer satisfaction is determined by an optimal combination of monetary value and performance. A customer has to get the best possible performance for the lowest possible price in order to be satisfied, buy and return to the e-service in the future. To achieve the best customer satisfaction an e-service must distribute its load over its child-services in an optimal way with respect to this criterion. We believe that we will obtain the best customer satisfaction with respect to the whole system if all nodes in the tree with a choice of child-services distribute their load in an optimal way.

Instead of reasoning about how to measure customer satisfaction and how to express it as a function that can be maximized we introduce a new, in our opinion more obvious term, that is the profit of the e-service. We assume that high customer satisfaction will lead to high profit.

In order to be able to talk about optimizing an e-service we have to formally specify the situation we are dealing with. We consider an e-service with several child-services out of which one has to be chosen for each transaction that needs to be run on a child-service. But at this point we do not take the view of an individual transaction, for which the optimal e-service has to be found. Instead we take on the perspective of the e-service that has to distribute its load over its child-services.

***AAD: plugged this in from deleted section. Needs to introduce model plus edit. Pick one of two. *** Rests from section 2: We add to the above model the notion of customers and suppliers. For each service, there are customers and suppliers. Customers may respond to QoS perturbations of the services they use. The service itself is a customer of its suppliers, and a supplier of its customers. The reasoning now is that QoS impacts business value if and only if customers change their behavior. This seems a reasonable assumption, if we ignore that contracts may exist between suppliers and customers, which explicitly state the costs of certain QoS degradations.

We model e-commerce interactions as a tree. Associated with each interaction there is an end customer who issues a transaction to the entry node in the tree. In turn, each node issues transactions to each children, unless it is a leave. If we take the perspective of a single node, it acts as a supplier to its parent(s) and as a customer to its children. Or, if one prefers: a node's children are its suppliers, and a node's parents are its customers.

If a node has interactions with *all* its children, we speak of an AND transaction, otherwise of an OR transaction. The OR transactions leave room for nodes to change their behavior: by choosing a different child node as partner it can influence its financial gains. In our model, we assume that a certain load coming in the root of the tree, which has to be distributed over competing child-services of each OR transaction, so that the maximum profit is made and the user/customer, to whom the result is passed on is satisfied. ***AAD end of three paragraphs to pick from ***

Let us assume there is an initial probability π_i for choosing service s_i and $\sum_i \pi_i = 1$, with $\pi_i = P(\text{executing } t_j \text{ on } s_i)$ where t_j is a transaction that could have been run on all other services $s_j, j \neq i$ as well.

A service s_i is for our purpose defined through a performance parameter and a monetary parameter for each transaction that can be executed on this service

$$(s_i, t_j) \quad \rightarrow \quad (g(s_i, t_j), P_{ij})$$

where $g : \mathbf{S}, \mathbf{T} \rightarrow \$$ is a function, that assigns a transaction and a service a monetary value, either the cost for executing this transaction, or the price for the item delivered through the transaction. P_{ij} is the observed performance for the execution of transaction t_j on service s_i.

In the example system we look at, we consider performance in terms of the delay - that is the execution time of transaction t_j on service s_i. The only service that can choose one out of a number of child-services is the car e-service.

The profit function $G : \mathbb{P} \times \$ \rightarrow \mathbb{R}$ gives a maximum gain for maximum performance and minimum monetary value. It is the metric of the system that we assume to translate into customer satisfaction when taking on a global view on the system. Profit is maximized when each service s finds optimal probabilities π_i according to which the load is distributed over competing child-services s_i such that the following sum is maximized.

$$\max \sum_i G_i(\$_i, P_i) \cdot \pi_i$$

$$\begin{aligned} &\text{subject to} \\ &\quad \text{load} = \text{given value} \\ &\sum_i \pi_i = 1 \\ &\quad P_i = f_i(\text{load}, \pi_i). \end{aligned} \tag{1}$$

The conditions are - beginning with the last one - that performance depends on the load, more specific it is given as a function of the load and the load on each child node equals the fraction π_i of the load on the parent node. So the performance in each child node is a function f_i of the load put on the parent node and possible values for the fraction pi_i. The second condition from the bottom is that all fractions must add to one, the total load. The first condition is that in a given system we assume the load to be known. This will not always be the case and then the optimization would give an optimum distribution on the child nodes as a function of the load.

We have made several assumptions here. The first one is that the profit function G can be expressed as a mathematical function that can be optimized (this means it is continuous? differentiable??) and even more that it is known. In a real system this might not be the case and we might first have to estimate the profit function based on transaction data before being able to do optimization. The second assumption is that there is a known relation between performance and load. It is known that load and performance of a system are related, but the precise nature of this relation often is unknown. Then it either has to be estimated by running and monitoring the system or some typical behavior is assumed.

Getting back to the full picture of the e-commerce tree, the top node must determine the fractions in which it passes its load on to its child-nodes. Then each child-node knows its incoming load and can compute the optimal way to distribute it over its respective child-nodes. This will give an optimal load distribution for all nodes in the tree and it should come close to an optimal path for each individual customer.

Example: Ignoring the dependence of the performance on the load, we can see in the gain function G an ordering function, that gives priorities to the services s_i based on performance and monetary value. A simple way to implement this ordering would be the definition of a matrix as shown in table 1

	$\$_1$	$\geq$	$\$_2$	$\geq$	$\$_3$	$\dots$
$P_1 \leq$	s_1		s_2		s_2	$\dots$
$\leq P_2 \leq$	s_2		s_2		s_3	$\dots$
$\leq P_3$	s_2		s_3		s_3	$\dots$
$\vdots$						$\dots$

Table 1: One possible ordering of the child-services

and to always pick the service that is top most and left most. ■

If we look at customers with individual constraints on performance and monetary value, those constraints have to be combined with the profit function regarded in this section. As a result we will have to find the optimal child-service for each customer. We then have to do an optimization that does not give us the ratios for distributing the load but a possibly prioritized set of child-services to which the transaction can be sent without violating the constraints.

5 Conclusion

Starting from the assumption that there is a relation between QoS and business metrics in e-commerce systems we have outlined how to express this relation in an e-service, called Q2B-e-service, which we have in part implemented. We have shown how to formally describe an e-commerce system under the focus of relating QoS and business metrics. We have done this by dividing the user-perceived tree up in subtrees that represent customer-service-supplier relations.

We have shown how XML messages can be extended to include a monetary value, so that the Q2B-e-service can operate on both the QoS and the business metrics. We have extended an existing management e-service in a way so that it displays QoS and business metrics together. This display gives an intuitively understandable picture of how well the system is doing.

It became clear that load distribution in an optimal way is an issue when talking about how to make use of the relation between QoS and business metrics. We have formalized the e-commerce tree as the user sees it so that an algorithm for optimal load distribution could be formulated.

The final problem we addressed is how to automatically find out when a system or an e-service in the system is no longer doing well and how to react to it. Based on a regression analysis of observed data thresholds for alarm signals can be established. What happens upon receiving an alarm goes beyond this work. Techniques for improving performance should be applied next.

Future work in this field can be done in different ways. First of all it would be a great advantage to replace the simulation of the Accidental Tourist by data from a real system. Then the correlation between QoS and business metrics can be quantified which will give us the information as to what extent the assumptions we have made can be proven right. In addition an implementation of the optimal load distribution can be evaluated in much more detail. Obtaining real data has some major obstacles. Not many companies like to release their data and monitoring of business metrics would require instrumentation of e-commerce software where it is currently not done.

However, even without real-world data the optimization as well as sending alarms can also be done based on an extended simulation. The simulation must include multiple customers at the same time in order to make load distribution a meaningful task.

References

[BBK00a] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating User-Perceived Quality into Web Server Design. Technical Report HPL-2000-3, Hewlett-Packard Labs, 2000.

[BBK00b] A. Bouch, N. Bhatti, and A. Kuchinsky. Quality is in the Eye of the Beholder: Meeting User's Requirements for Internet Quality of Service. Technical Report HPL-2000-4, Hewlett-Packard Labs, 2000.

[BC99] P. Barford and M. Crovella. Measuring web performance in the wide area. *Performance Evaluation Review*, August 1999. Special Issue on Network Traffic Measurement and Workload Characterization.

[cha] http://www.internetsolutions.enterprise.hp.com/chai/-products/platform/chai_appliance.html.

[esp] http://www.e-speak.hp.com/.

[Fir] http://firehunter.comms.agilent.com/home.htm.

[Hil93] A. Hiles. *Service Level Agreements – Managing Cost and Quality in Service Relationships.* Chapman & Hall, 1993.

[Kan93] Kant. *Introduction to Performance Modeling.* Addison-Wesley, 1993.

[MB00] Lee W. McKnight and Jahangir Boroumand. Pricing internet services: Proposed improvements. *Computer*, March 2000.

[MDGH99] V. Machiraju, M. Dekhil, M. Griss, and J. Holland. E-service Management Vocabulary: Conceptual and Reference Guide. Internal document, November 1999.

[MDGW00] V. Machiraju, M. Dekhil, M. Griss, and K. Wurster. E-services Management Requirements. Technical Report HPL-2000-60, Hewlett-Packard Labs, 2000.

[Obs] http://www.openview.hp.com/products/webtransobserver/.

[PJA+00] G. T. Paixao, W. Meira Jr., V. A. F. Almeida, D. A. Menasce, and A. M. Pereira. Design and implementation of a tool for measuring the performance of complex e-commerce sites. In B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, editors, *Proc. 11th Int. Conf. on Computer Performance Evaluation - Modelling Techniques and Tools (TOOLS 2000)*, volume 1786 of *Lecture Notes in Computer Science*, pages 309–323, Schaumburg, IL, USA, March 2000. Springer.

[SLA] http://www.dmtf.org/info/sla.html.