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

Katinka Wolter[1], Aad van Moorsel
Software Technology Laboratory
HP Laboratories Palo Alto

HPL-2001-96
April 23rd , 2001*

e-services, QoS, QoE, QoBiz, management, business metrics, alarms, optimization, monitoring

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] TU Berlin, Sekr. FR 2-2, Franklinstr. 28/29 10587 Berlin, Germany

To be presented at the Open View Forum, New Orleans, LA June 11-15, 2001.

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

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

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

April 7, 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. In this paper, we denote this relationship as the 'Q2B' relationship ("quality of service to business"). For IT and business managers there is thus 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 system 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.

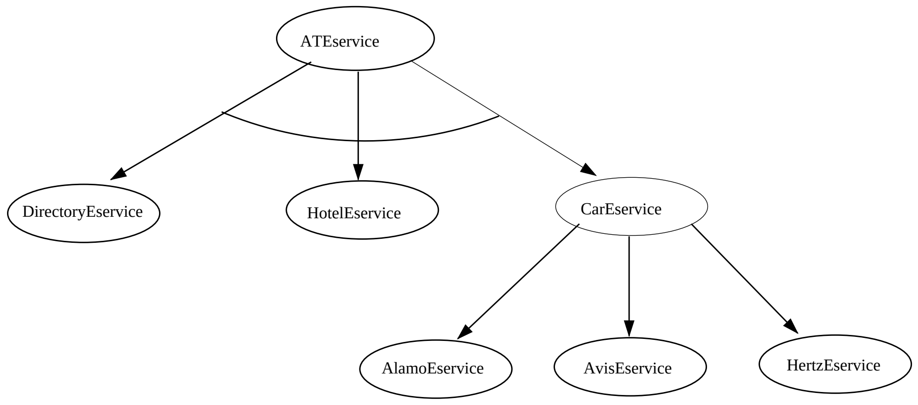

Figure 1: Accidental Tourist services configuration

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 serves some administrative purpose. 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. Moreover, 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 systems [PJA$^+$00, BC99, Obs, Fir], but existing work in system management rarely considers the business impact of QoS. If business aspects are considered this is often based on service level agreements (SLAs) [SLA, Hil93]. SLAs, however, provide a *static* relationship between QoS and business impact; that is, customers pay at various QoS levels corresponding to some predefined contract. 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 influencing the business results of e-services. An important tool for e-service providers to counter balance the customer behavior is pricing [MB00]. We will briefly touch on pricing in what follows, but mainly concentrate on more elementary aspects of visualizing and exploiting the Q2B relationship.

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 (affirmative) 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].

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 individual e-services are simulated as random delays between consecutive messages. Instrumentation interceptors are plugged in the e-speak middleware [Pru00]. They are used to monitor response times and send them to the management e-service.

Several interceptors can be used for monitoring different features - one interceptor for each feature. They can be located on the managed service's side or on the management service's side. In Figure 2 this is illustrated with two interceptors on each side.

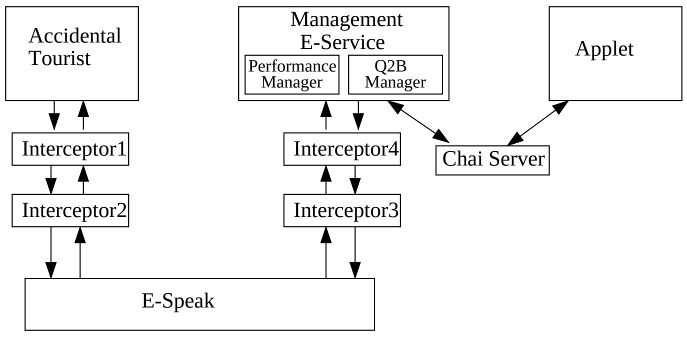

Figure 2: Structure of the management system

The management e-service receives on one hand data from the accidental tourist simulation and on the other hand requests from different user applets (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 metrics. In the management system prototype, there already exists a performance manager, to which we added the Q2B manager. The applets and the management e-service communicate through the Chai server [cha].

To close this section, Figure 3 introduces in some more detail the control flow of the accidental tourist simulation. The accidental tourist processes one request at a time. The first path of transactions is created by running *call-for-proposal* transactions on all but the directory e-service, which does

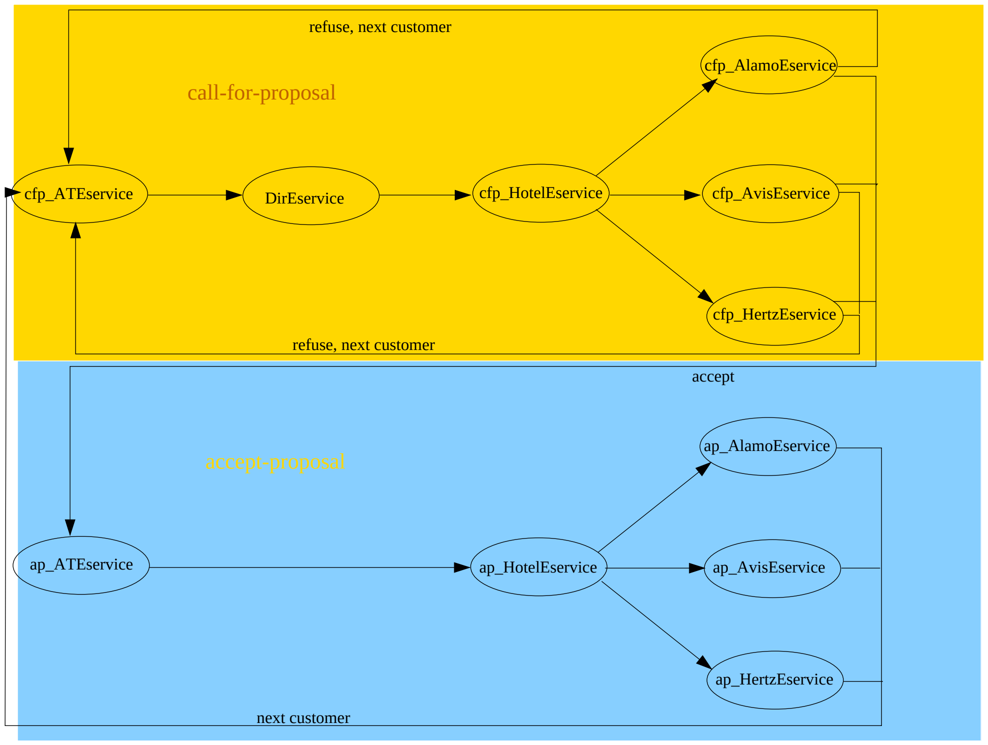

Figure 3: Flow Chart of the Accidental Tourist

some administrative jobs. Then the proposal is accepted with a certain probability (the arc labelled *accept*) and refused otherwise. If 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 after the *accept-proposal* transaction on one of the car e-services or after the proposal is refused.

2.2 Adding Business Metrics to E-Service Management XML

The e-service management system receives data using XML messages. It either receives these messages 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* (note that we use terminology such as services and transactions according to [MDGW00]). 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. 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 to indicate it happens only once, at initialization.

At run time, through its interceptors, 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*, which the management e-service computes from the system information.

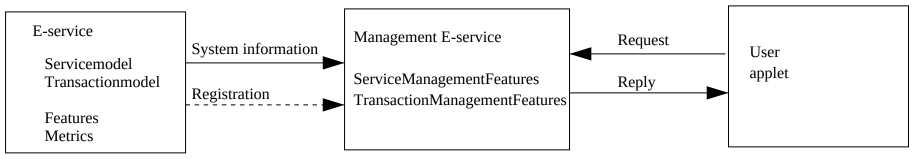

Figure 4: XML contents and message flow

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 the optimization set-up in Section 4, 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 and features exist for transactions, we introduce a new **Feature**, of type `Monetary`, as follows:

Feature

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

Revenue denotes the money value attached to a successful transaction. This can also be a cost, fee, or gain, depending on the semantics of the transaction. Loss on the other hand is the money value associated with a failed transaction.

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. Metrics are used in two ways: when an e-service sends its data 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. The latter is not part of the shown message. 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. Formally this message does not use the metric statement, but the transmitted data corresponds to elementary metrics. Note that we only show part of a much more extensive message that includes 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 Kiviat graphs (using terminology from Jain [Jai91]–these graphs are also known as 'radar plots') 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 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, and then assume that customers will choose a competing service instead. Note that one would like to be able to demonstrate that this is

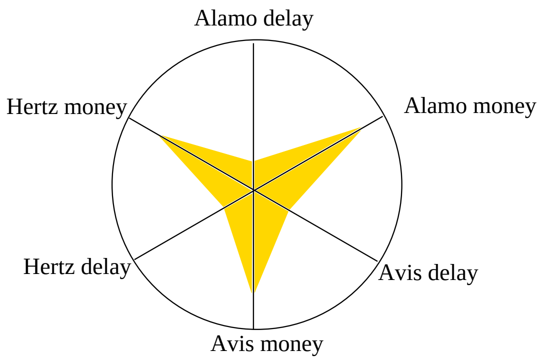

Figure 5: Desired shape of Kiviat graph

according to the operation of real e-commerce systems, and we refer to [BBK00a, BBK00b] 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 Kiviat graph gives an intuitively understandable picture of whether a system is operating well or not. More specific, it shows how QoS and business results correlate; they are a first step towards finding the root cause for changing business results.

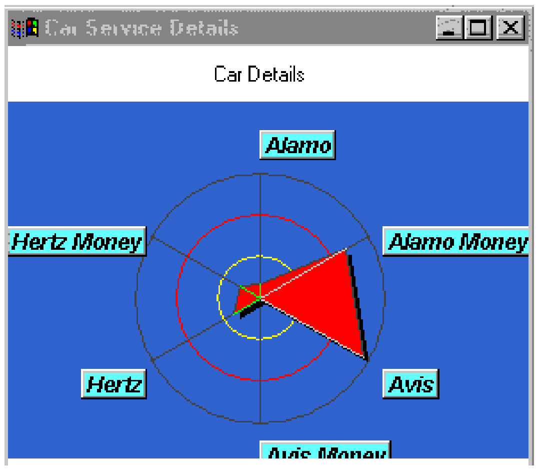

Figure 6: Screen shot of Kiviat graph

A screen shot of our implementation as Avis' performance is turning bad is shown in figure 6. 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 of the interrupted transaction 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 instead of making a random choice. 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 Section 4.

The counters in Figure 7 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.

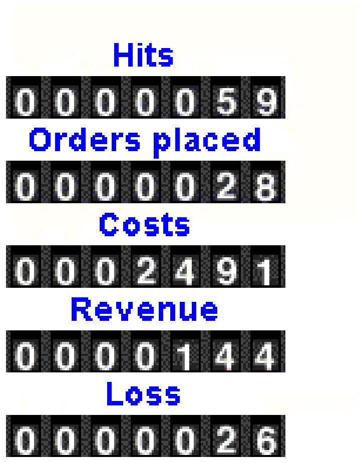
Hits
0000059
Orders placed
0000028
Costs
0002491
Revenue
0000144
Loss
0000026

Figure 7: 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, and see if QoS influences this health parameter. 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 Kiviat graph 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 threatens to cause loss of revenue.

3 QoS-Based Alarms for Business Degradation

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 QoS 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 various ways; for instance, 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.

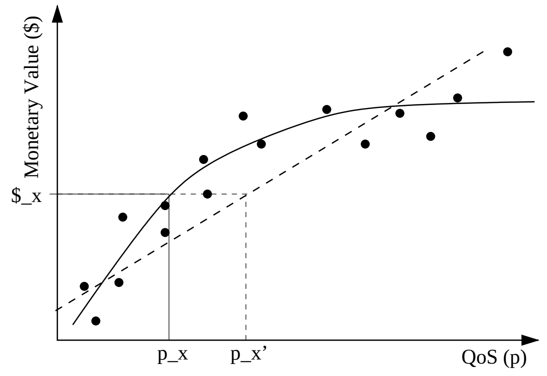

Figure 8: Estimation of performance and monetary value correlation

Figure 8 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 8 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 8. 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 can be based on performance values, business values or parameters of the correlation curve. Figure 8 illustrates one 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 this is marked by monetary value $\$_x$. Then one finds the corresponding QoS value using the correlation curve. This result in value p_x (or p'_x, depending on the used correlation curve). Now, if samples smaller than p_x occur, an alarm will be sent. Details of this procedure must be fine tuned depending on the particular situation.

Alarms based on the value of the business metric may be used to indicate 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 triggered 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.

4 Optimizing E-Service Business Operation

In this section we discuss how to optimize the business return when QoS perturbations influence the business revenue. From the perspective of an e-service provider, when the Q2B relationship is well understood, it allows one to respond to alarms and configure the system based on either QoS or business measurements. Which enforcement capabilities are appropriate is highly dependent on the particulars of the infrastructure. For instance, computing power may be added to a server cluster, or priority mechanisms of load balancers can be adapted. Understanding the Q2B relationship gives the business manager tools to determine what investment in IT is most appropriate for the business.

As pointed out in the introduction, very complex dynamics exist among e-services and its customers. We formulate an optimization problem for one particular setting. In this optimization problem we take the perspective of an e-service optimizing the amount of transactions to send to its suppliers. We assume that the considered e-service is a prime customer of its suppliers, and its load influences the QoS delivered by the suppliers. One can imagine situations like this when brokers have a dominating central function, distributing customer orders between various suppliers.

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 its children, unless it is a leaf. 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 supplier it can influence its financial gains. We assume that a certain load 'enters' the root of the tree, and is then distributed over competing child-services of each OR transaction. The optimization problem is to determine what fraction of load to send to each of the competing child services.

Formally, at a node s, there exists an incoming load $\texttt{load}$, and a fraction π_i of the traffic goes to the ith child node s_i, $i = 1, \ldots, n$ ($\sum_i \pi_i = 1$). Let \$ be the monetary return for node s. We want to optimize \$, which is a function of P, the QoS at node s (say, $\$ = \psi(P)$.) Since the QoS of a node depends on its suppliers, we get that P is a function of $P_i, i = 1, \ldots, n$, say $P = f(P_1, \ldots, P_n)$. Since we assumed that the load influences the QoS of a node, we also have a relationship between P_i and $\pi_i, i = 1, \ldots, n$, as follows: $P_i = g_i(\pi_i \times \texttt{load})$. A mathematical programming formulation then becomes:

$$\begin{aligned} \max \$ = \psi \circ f(P_1, \ldots, P_n) & \\ \text{subject to} & \\ \sum_i \pi_i = 1 & \qquad (1) \\ P_i = g_i(\pi_i \times \texttt{load}) & \end{aligned}$$

To recap, in the above formulation, the function f relates the performance of the children to that of the parent node, and ψ relates parent node performance to business value.

It is interesting to note that the above formulation is not only relevant for the broker scenario we described. It also holds if we discuss the optimization of a load balancer that picks between databases or application servers (node s would then play the role of load balancer). This illustrates the opportunities that open up once one is able to establish a statistical relationship between QoS and business metrics. Old-fashioned IT optimization then gets replaced automatically by business optimization.

Acknowledgments

We thank Vijay Machiraju for assisting in adding the Q2B e-service to the e-service management system. We thank Fabio Casati for feedback and comments (albeit some in Italian), and Pano Santos and Alex Zhang for discussing and help in clarifying the optimization opportunities.

5 Conclusion

In this paper we introduced a management system that deals with QoS and business metrics combined. This is an issue of increasing importance, since e-commerce and other Internet business applications blur the distinction between technology and business. This paper touched on several important issues, particularly those related to the financial consequences of QoS perturbations. In particular, we introduced the Q2B e-service that monitors QoS and business metrics, sends alarms when QoS threatens to influence the bottom line, and optimizes supplier choices based on QoS considerations.

Obviously, we do not know exactly what the future of business e-services will be, and what kind of instrumentation will be available. However, we have demonstrated in this paper the opportunities that open up if one is able to statistically characterize the Q2B relationship. The information gained turns IT management directly into business management.

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.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis.* Wiley, 1991.

[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.

[Pru00] J. Pruyne. Enabling QoS via Interception in Middleware. Technical Report HPL-2000-29, Hewlett-Packard Labs, 2000.

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