End-To-End Response Time: Where to Measure?

Dr. Tim R. Norton
Simalytic Solutions, LLC
CMG99 Session 423, December 8, 1999

Because of the increasing need for application measurement, several vendors are offering products to track and record End-To-End Response Time. The question now becomes "where to measure" instead of "how to measure." Measuring application response time at several points can produce an overwhelming amount of data but does not guarantee any meaningful information from the business point-of-view. This paper discusses different approaches to selecting these measurement points and presents several hypothetical examples to illustrate their meaningfulness to business transactions.

1. Introduction

When looking at transaction based applications, it is easy to focus on the methods of measurement that are well known and comfortable. However, these don't provide enough information to understand what is happening at the business level of an application with components spread across multiple heterogeneous systems. The argument has been made that business transactions will meet the overall service objective if each of the components meets a service objective for that component. This approach becomes inadequate when applications start sharing common functions or when the application is complex enough that different functions have different service objectives (i.e. query versus update).

New applications are no longer batch systems on a single computer. They are now multi-platform on-line transaction processing client/server systems combining departmental servers and mainframe repositories. Such complex application designs utilize the features and services of different types of computers (mainframe, mid-range and desktop) and often impact several aspects of the overall business. Techniques to measure individual systems have been well understood for some time. Techniques to measure applications have gained sophistication and popularity over the last several years. The challenge now is to provide a business focus that gets the best return on application measurement while reducing the cost of data collection.

This paper doesn't present "the solution" because that are no hard-and-fast *right* answers. What it does present is a look at the relationship between the business aspect and the technology aspect of measuring an application. The value of limiting data collection to what is needed to address the business problem is also discussed. *Section 2, Response Time* Measurement, presents a brief overview of measuring application response time; *Section 3, Transaction Measurement,* discusses the differences between measuring Information Technology (IT) transactions and Business Transactions; and *Section 4, Hypothetical Applications,* illustrates how the measurement points (where measurement data can be collected in an application) impact the business/application relationship using hypothetical applications.

1.1 Background

Today's computer environments must be viewed with the objective of understanding how the systems meet the end user's requirements. By addressing the business needs, we avoid viewing the computing environment as an end in itself and relate the benefits of an application to the business that depends on it. Unfortunately, there are many pressures that try to shift the focus to advances in technology without regard to what the computing environment provides to the business it was intended to support. Advances in technology lead to a very rapid change, which means it is often difficult to relate the value of many of the changes to the overall business in an objective manner. (Business in this context means more than a for-profit company. It can include any type of company, institution, agency, or organization with an overall objective, be it revenue, service or regulatory.)

Measurement has traditionally focused on resources (CPU utilization, I/O rate, etc.) and workflows (job throughput, internal transaction response times, etc.) to determine if a given system is "good enough" to service a workload (which, in theory, translates to an application). Today, application measurement in large computer installations with multiple systems requires an understanding of not only the operating systems, the platforms, the clients, the servers, the networks, the transaction systems, etc., but also the relationships between them and the business objectives (such as staffing levels and “widgets” sold). This relationship allows business managers to understand the impact of application responsiveness on the overall business objectives. Instead of analyzing individual systems, the responsiveness of the application needs to be understood across the entire enterprise to insure that the computing environment addresses the requirements of the business objectives and goals. But this understanding requires measurement of both the application (end-to-end response time) and the individual components (internal response time).

2. Response Time Measurement

The idea of response time measurement has been around for a long time. Early mainframe transaction systems, such as CICS and IMS, quickly developed robust measurement facilities. Often referred to as *internal* response time, these measurements reflect the time from when the host system receives the transaction until a response is sent back to the user. The application was considered to be performing well if the internal response time was within prescribed limits. Network time was generally considered a separate (external) problem to be dealt with by the network support organization.

The proliferation of multiple-tier client/server applications has made response time measurement much more complex. There is no longer a single place to collect the measurement information needed to determine how an application is performing. Network time is now interwoven with the response times of other components of the application and cannot be easily deferred to another organization. Furthermore, even if all the transaction data is collected, it often doesn’t contain enough information to understand the impact of the application’s performance on the overall business.

It is now much more important to look at application response time from a business point-of-view. The major business related benefits of using application end-to-end response time measurement are:

- **Service Level Agreements** – evidence of compliance to Service Level Agreements that have been formally negotiated between the IT organization and the end users.
- **Application Prediction** – predicting application response times at increased loads for capacity planning or Service Level Agreement modification.
- **Configuration Management** – understanding the return on investment to the business for changes in the physical environment such as server upgrades or network enhancements.

These benefits are focused on the business objective, but each can be expanded to include the more traditional Information Technology activities. For example, Service Level Agreements can include day-to-day performance activities, and Configuration Management can include capacity planning.

2.1 Current Techniques

The details of various techniques for measuring response time are available in the current literature (Knight and Haworth 1996; Lipovich 1997; McBride 1997; Ramanathan and Perry 1999; Smead 1998; Smith and Williams 1998; Thompson, Muñoz, and DeBruhl 1997; Tsykin and Langshaw 1999). Some of these concepts are briefly defined below. The interested reader is encouraged to refer to these, and other, papers on various aspects of the general topic of application response time measurement for specific implementation details.

Tsykin and Langshaw (1999) provide a good overview of the general techniques. They list four broad techniques for application measurement:

- Application instrumentation: modifying the application at the source code level to collect performance data.
- Client instrumentation: inserting hooks into the client environment to collect data on activities such as operating system interrupts and/or messages (as with Microsoft Windows).
- Wire Sniffing: monitoring, decoding and analyzing either raw network (sniffer) traffic or server network packets (i.e., TCP/IP).
- Benchmarking: application scripts periodically executed and measured.

Although there are variations of these techniques in both the cited references and other sources, the general concepts fall into the four broad categories above. There are a very large number of factors involved in any decision to measure an application. Because each of these authors has focused on a somewhat different combination of these factors, their conclusions, and suggested solutions, are more focused toward one of the four categories than the other three.

Tsykin and Langshaw (1999) are concerned about the volume of data and processing required to correlate individual units of work across a complex enterprise, so they advocate a technique based on client instrumentation that characterizes user work patterns, instead of collecting detailed transaction data.

McBride (1997) focuses on the need to identify and measure the business transaction and advocates the use of application instrumentation with ARM (Application Response Measurement).

Smead (1998) provides an in-depth analysis of different application instrumentation techniques, but is focused on the low-level IT transaction, rather than the high level business transaction.

Lipovich (1997) takes an application level, rather than resource usage, view but looks at the compo-

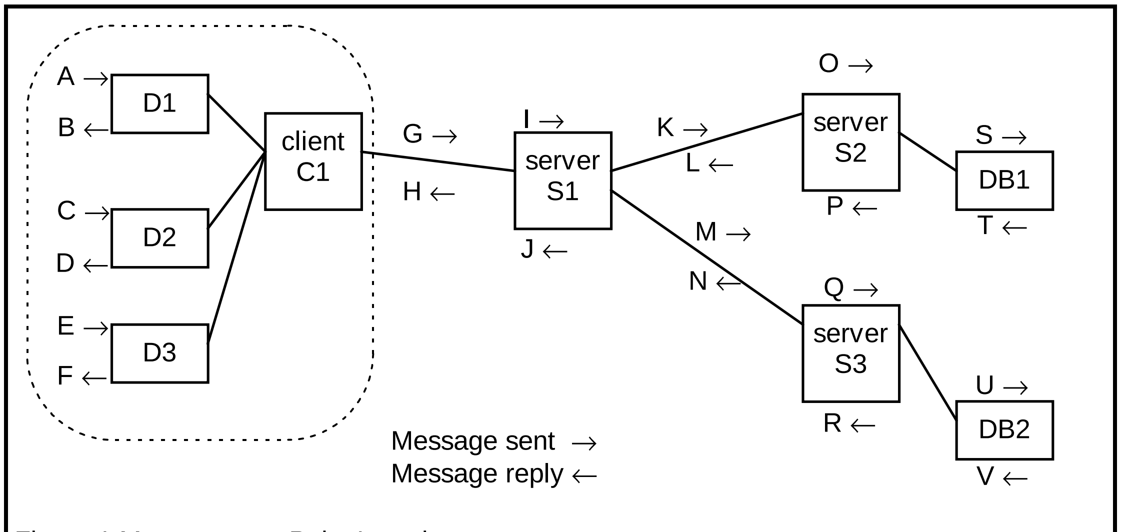

Figure 1 Measurement Point Locations

nents of the application response time from a server-centric perspective.

Smith and Williams (1998) provide an interesting approach to understanding application design for modeling with the use of Message Sequence Charts to describe application behavior. These charts provide a very clear representation of how messages and functions flow between servers or components in an application. Their use of these charts to represent three types of CORBA-based synchronization (synchronous, deferred synchronous and asynchronous) highlight some of the pitfalls in selecting measurement points and attempting to correlate resource usage to end-to-end response time.

2.2 Measurement Points

Each of these is a valid approach to a given measurement problem and illustrates the real issue of understanding the need before selecting the solution. Many approaches to end-to-end response time measurement are based on a bottom up approach – look at the data that can be collected and figure out what can be done with it that is meaningful. This author believes a top down approach is more productive for the overall business. By starting with the business needs and functions, the measurement effort can focus on what supports business improvement and will be more quickly accepted by management and other sponsors. In addition, because there is a direct correlation to business value, the data collected can be easily identified as either required or unnecessary.

Figure 1, Measurement Point Locations, shows a graphical representation of a client/server environment with many of the common measurement points labeled. The letters A through V represent measurement points with an arrow showing the direction of transaction flow. The other elements are dialog boxes[1] (D1, D2 and D3), the client workstation (C1), servers (S1, S2 and S3) and databases (DB1 and DB2). To avoid an overly complicated figure, there is no distinction between the client and the application measurement points at C1. Measurement points A and F are used to represent both, depending on which other measurement point they are paired with. Using A as the transaction start and B as the transaction end, (A-B) represents the client instrumentation that measures a single dialog box or screen. Using A as the transaction start and F as the transaction end, (A-F) represents the application instrumentation that measures a business transaction. Server based instrumentation is really a form of internal response time measurement that has been available for years. Although the implementations differ, there is no real conceptual difference between measuring a transaction at a middleware server such as S1 (I-J) and measuring it at a legacy host server such as S2 (O-P).

The reader is reminded that this is a symbolic representation and not intended to be a definitive detailed description. Each real client/server application will differ. In addition, the author cannot be familiar with every available end-to-end measurement tool, some of which may implement additional measurement points. The figure provides only enough detail to provoke thought and discussion. The purpose of this paper is to encourage critical thinking about what is being measured and where the measurements are taken (measurement points). It is expected that readers will identify additional measurement points supported by tools with which they are experienced. Hopefully, this look at response time measurement will encourage them to ask why these measurement points are needed and what the resulting metric means to the business. This is not intended to imply any negative connotation to these, or any other, measurement

[1] The term “dialog box” is used here because most of the client instrumentation tools are specifically designed for instrumenting Microsoft Windows environments. The concept applies equally to whatever user interface the application uses for screen formatting.

points, but to encourage the readers to use the information presented here as a starting place from which to develop the relationship between the transactions and the resulting business value of a given application.

2.3 Using the Techniques

The four techniques for application measurement (Application Instrumentation, Client Instrumentation, Wire Sniffing and Benchmarking) can be described using the measurement points identified in *Figure 1*.

- **Application Instrumentation** – Instrumenting the application means measurement events are generated by the application at the start (A) and end (F) of a meaningful business transaction. If the application creates measurement events for only the start (A) and the end (B) of the business transaction, the end-to-end response time may include user think time (B-C and D-E). The value of user think time is very application dependent and may, or may not, be appropriate to include in the end-to-end response time. For example, it would be reasonable to include user think time when calculating the calls per hour for the operators in a call center to determine the number of operators needed but not to determine the impact of a server upgrade.
- **Client Instrumentation** – Instrumenting the client means measurement events are generated by "hooks" inserted into the client environment (generally by a purchased tool) at the identified API (Application Programming Interface) between the application code and the client operating system environment. Microsoft Windows has a rich, well documented API which accounts for the growing number of tools designed specially for that environment. This approach is limited by the instrumentation tool's ability to differentiate between multiple instances of the same dialog box within a single business transaction. If the tool cannot distinguish any difference in the measurement points that terminate the transaction (B, D or F) then this approach will measure multiple Information Technology transactions (A-B, C-D and E-F) instead of a single business transaction (A-F). If it can decode the data in the API call, then it is possible to distinguish the intermediate measurement points from the final measurement point and identify the business transaction (A-F). User think time is also an issue with this approach. In addition, the very large number and complexity of API calls in an environment such as Microsoft Windows makes the task of inserting the "hooks" very tedious and susceptible to error.
- **Wire Sniffing** – Instrumenting network segments ("the wire") between clients and servers is very non-invasive because it does not require any modifications to either the application or the client environment. This approach provides very good data about network utilization and traffic. It can also provide application topology information by matching sending and destination packet addresses. Other application information may be available if the instrumentation software (either real-time or post-processor) has enough knowledge to decode the application level packet; however encryption can seriously limit this capability. This approach can measure response time for a packet or message (G-H, K-L, or M-N), but it can do little to measure the response time of the business transaction. Even with sophisticated matching and correlation, the client processing component and the network time between the client and the network measurement point (A-G and H-B) are not captured in the measurement.
- **Benchmarking** – Instrumenting by benchmarking (sometimes referred to as synthetic transactions) means to create a script or otherwise controlled execution of selected functions in an application. The script is run automatically at an interval that will provide a reasonable sample of the application response time without adding a noticeable server or network load. The main advantage to this technique is that script response time can be more accurately measured than the actual production environment of the application. Benchmarking can be implemented using a variety of techniques, including application instrumentation or client instrumentation. The issues discussed above for those approaches still apply when they are used for benchmarking. In addition, because it is actually a sampling technique, it only provides an approximation[2] of application response time. A further problem with this approach is that it often measures the application over a limited subset of the network. This can cause the response time measurement to be skewed (either too low or too high). The benchmark must be very carefully designed, and constantly reevaluated, to avoid this problem.

Each of these techniques has advantages and shortcomings. There is a direct correlation between the ability to measure non-invasively and the ability to measure application functions. The greater the application understanding, the more invasive the measurement technique becomes.

[2] A discussion of statistical sampling is beyond the scope of this paper other than to note that care must be taken when selecting the interval and when summarizing the measured results. The interested reader should refer to a college level statistics textbook for additional information about these issues.

3. Transaction Measurement

For any given application, understanding the measurement points (where measurement data should be collected in the application) is a function of both what the application does and the objective of the measurement. What constitutes a transaction? How do we count them? What is the business impact if there are more (or fewer) transactions than expected or if they take more (or less) time to complete? Measurements for Service Level Management (Service Level Objectives and Service Level Agreements) may satisfy a political need, but they will be frustratingly useless if they do not provide enough information to determine what is causing the application response time to fail to meet the service objective. On the other hand, large amounts of resource-centric information (i.e., CPU utilization, network segment utilization, I/O rates, etc.) doesn't help the decision makers understand the impact to the application at the business level.

End-to-end response time measurements must be related back to the business impact. Response time is a valid metric only if there is some valid business reason to use it. An example of this relationship is shown by relating the responsiveness of an application supporting an order entry call center to the number of calls an operator can handle in an hour. The fewer calls the operator handles means the more operators that are required for a given call volume. That relationship provides the information necessary to make the business decision of upgrading the server or hiring more operators (Norton 1998).

3.1 What vs. Where

When collecting end-to-end response time data, the most important question is "Where to measure?" The nature of the application and the business should define what to measure. The Information Technology organization's task is to figure out how to collect that measurement data. The decision will almost always come down to measuring either Business transactions or Information Technology transactions depending on the objectives of the measurement activity.

3.2 Understanding IT Transactions

What are IT Transactions? Although there is no formal definition, these are the transactions we most often think of from an Information Technology perspective. These are the transactions that are reported by the built-in data collection included in most current operating systems, transaction systems and databases. The best examples come from the MVS (OS390) environment and include transaction based systems (such as CICS, IDMS, and IMS), databases (like DB2 and Adabas) and even the operating system itself (measuring jobs and TSO sessions).

The open systems and NT environments are rapidly closing the "measurement gap" by providing measurement data from most of the major transaction based systems (such as CICS6000 and Tuxedo), databases (like DB2, Oracle, Informix and Sybase) and other components (such as MQSeries and SAP). Generally these transactions are identified and measured on individual servers with little correlation to activities on other servers. The response time measurement most often associated with Information Technology transactions is internal response time because it is the measure of response time internal to the particular system without other components such as the network. (For example, the CICS response time measurement refers only to time the transaction spends inside CICS). Internal response time is usually measured by the transaction or database system.

An increasing complication with these systems is their attempt to account for the response time component outside of their environment. Although much more complex in today's client/server environments, it is the same problem we have always faced when dealing with spawned transactions and complex database queries. The response time of a transaction includes both processing time and wait time. But wait time is really the response time of another service or component, which is also both processing time and wait time. Correlation, rather than collection, is the issue.

3.3 Understanding Business Transactions

What are Business Transactions? Again, there is no formal definition, but these are the transactions we most often think of from an end user's perspective. A Business Transaction is much harder to define in general terms because it completely depends on the nature of the business. A mail-order business might define business transactions as placing an order over the phone and shipping a package to a customer. A restaurant might define a business transaction as serving a meal to a group of people. In either case, the end user has a relatively clear understanding of a transaction, but there can still be differing perspectives. Does the restaurant transaction start when the customer walks in the door, when they sit at the table or when the server greets them? It depends on who is doing the measuring and what their objective is in doing it.

3.4 The IT to Business Relationship

The overall objective is to develop a relationship between the business transactions and the Information Technology transactions. Benchmarking and synthetic transactions (Ramanathan and Perry 1999) provide a compromise between internal response time measurements and true end-to-end response time measurement because they are typically much closer to the application services. However, they bypass much of the network experienced by the end users. The results of these techniques are quite valuable when combined with true end-to-end response time to identify network performance issues. These performance issues are often dismissed because it is thought a waste of time and effort to measure what is outside the control of the application, but they have a direct impact on the overall business.

Knowing the end-to-end response time isn't enough if the application uses common services shared across several applications. For example, Ramanathan (1999) measures Network File System (NFS) services and assumes an average "good" response time is good enough for all users of the service. However, it is easy to envision two applications using the same NFS service with very different I/O characteristics and response time requirements. We often need to measure both the end-to-end response time and the service response time. The difficulty is coordinating the two measurements to understand how changes in the service response time impact the applications at the business level (i.e., which application transactions are responsible for which server transactions).

In the final analysis, the measurement objective has the greatest influence over the type of transactions to measure. Business related questions will focus attention on business transaction measurement and resource related questions will focus measurement on IT transaction measurement. While it is certainly possible, and even desirable, to collect both measurements, any correlation between the two must be done very carefully. Issues that are mutually exclusive, such as user think time, can cause the correlated results to be misleading.

4. Hypothetical Applications

This section presents several hypothetical applications, starting from the simplest, and discusses how each of the four broad techniques for application measurement presented in *Section 2, Response Time* Measurement*,* (Application Instrumentation, Client Instrumentation, Wire Sniffing and Benchmarking), can have different business level impacts. Other issues regarding measurement points shown in *Figure 1* will also be discussed as appropriate.

The applications presented in this section are completely hypothetical and are meant only to illustrate ***some*** of the approaches to client/server application development and identify ***some*** of the issues with measuring each approach. The author makes no claims regarding the completeness of the list or the likelihood that any listed technique is actually used. It is not intended to be an exhaustive list of client/server implementation techniques. It is intended to provide the reader with some insights into how to view application measurement from both business and IT perspectives. References to real applications are solely for the purpose of illustrating the technique and are based purely on the author's speculation of how those applications have been implemented.

4.1 Simple Web Application

One of the simplest examples of a client/server application is a single Internet web page used to front-end an existing application inquiry. It is very easy to measure the business transactions by measuring either the web transactions or the IT transactions because there is a one-to-one correspondence between them.

This application would be implemented with only some of the components shown in *Figure 1.* A single page web browser form, D1, is used by the user to request information. There is a simple mapping of the web transaction (A-B), implemented as an HTML page invoking a *cgi* script at web server S1 as I-J, to the IT transaction, implemented as a legacy transaction using CICS or IMS at S2 as O-P. The response time of database transactions would be measured by S-T. A real example of this type of transaction is the FedEx web page to track the shipment of a package (http://www.fedex.com/us/tracking/). This single web page uses a *cgi* script with in-house developed middleware to invoke an IMS transaction. Each tracking number inquiry maps directly to a single invocation of an IMS transaction.

Benchmarking is a reasonable approach if the sampling and network subset issues can be adequately addressed. Application instrumentation and Client instrumentation both provide about the same level of response time information because there is little practical difference between the web transaction and the business transaction. Wire Sniffing could also provide good information because the application is straightforward and it should be relatively easy to match the start and end network messages. All of the techniques must be implemented at multiple measurement points (i.e., I-J, O-P and S-T), and the results correlated, if any of these components are used by other applications (or other types of transactions in the same application).

4.2 Multi-Page Web Application

A more complex example of a client/server application is using multiple Internet web pages (D1, D2 and D3) to front-end an existing application inquiry. It is no longer possible to measure the business transaction (A-F) by measuring the web transactions (A-B, C-D, and E-F) because there is not a one-to-one correspondence between them. Several web browser pages are all part of a single form. Most of the real work happens when the submit button on the last page is pushed, causing that transaction (E-F) to have a significantly higher response time. The multi-page web form could invoke multiple IT transactions, such as customer look-up (O-P) followed by order look-up (Q-R). Examples are the Travelocity form to select airline flights (http://dps1.travelocity.com:80/airgrqst.ctl) and the FedEx package tracking page to check the status of multiple packages at once (with multiple invocations of the same IT transaction).

Benchmarking is a reasonable approach if the sampling and network subset issues can be adequately addressed. Application instrumentation provides the best measurement of the business transaction (A-F) but now user think time (B-C and D-E) is an issue. Client instrumentation provides a business transaction view (A-F) only if the tool can distinguish the end of the transaction (F) from the end on the intermediate screens (B and D). Wire Sniffing could also provide good information because the application is straightforward, and it should be relatively easy to match the start and end network messages, although multiple pages will increase the number of messages and thus the complexity to match them correctly. All of the techniques must be implemented at multiple measurement points (i.e., I-J, O-P and S-T), and the results correlated, if any of these components are used by other applications (or other types of transactions in the same application). Another issue that now must be resolved is overlap in the sub-component response times. Does the server S1 perform the tasks in series or in parallel? In other words, is the response time I-J the sum of K-L and M-N or the longer of the two? Although this question doesn't impact the data collected at C1, it does have a great deal of influence on how that end-to-end response time is interpreted.

4.3 Complex Web Application

One to several web pages (D1 and maybe D2 and maybe D3) request services from multiple hosts (middleware server S1, database server S2 and transaction system S3). The exact processing in each step varies depending on the results of prior requests. The customer sees a business transaction as a single web function which may take more than one screen or page (A-F), but it requires several web transactions (e.g., cgi scripts, I-J), each of which invokes one or more IT transactions (e.g., IMS and DB transactions, O-P and Q-R). A real example of this type of transaction is the FedEx web page to ship a package (https://www.fedex.com/cgi-bin/ship_it/interNetShip?us). This web page uses *cgi* scripts in conjunction with in-house developed middleware to invoke an IMS transaction to check the status of the customer number provided by the user, followed by other transactions to other client/server application servers and to web database servers.

Benchmarking is still reasonable but much more complex even if the sampling and network subset issues can be adequately addressed. The complexity of the transactions means the scripts must also be very complex, which greatly increases the probability that a significant scenario will be overlooked. Application instrumentation provides the best measurement of the business transaction (A-B or A-D or A-F), but user think time (B-C and D-E) is still an issue. Client instrumentation provides a business transaction view (A-B or A-D or A-F) only if the tool can distinguish the end of the business transaction from the end on the intermediate screens, which is now more complex because it can be the end of any of the dialogs (B, D or F). Wire Sniffing is now extremely difficult because the application is not straightforward. The complexity to match the start and end network messages quickly becomes overwhelming, although there are some tools that claim to have the required level of sophistication. All of the techniques must be implemented at multiple measurement points (i.e., I-J, O-P and S-T), and the results correlated, to understand the actual topology of the application. The input data dependent nature of the application behavior greatly increases correlation of the data from these different measurement points. The issue of overlap in the sub-component response times increases with this level of application complexity.

4.4 COTS Applications

A COTS (Commercial Off The Shelf) application provides a set of services implemented on middleware servers, desktop clients or both. The middleware component may also invoke transactions from other services, such as legacy host databases or transaction applications. Most of the issues are the same as stated above in *Section 4.3* for a *Complex Web Application* but the commercial (and thus proprietary) nature of these applications almost completely eliminates the possibility of Application instrumentation. Although many COTS vendors have announced support for the ARM API, there has been little effort to coordinate the techniques used for transaction identification across multiple servers. This means that the ARM data from the desktop cannot be correlated with the ARM data from the middleware servers. Even if the desktop client is in-house written, the COTS middleware component may not accept a transaction identifier. There has been some success with "wrappering" the application (adding in-house written code both before and after the COTS code to generate the measurement event), but it is questionable if this technique is significantly more effective than Client instrumentation.

4.5 Object Request Brokers

Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA) and Microsoft's Distributed Common Object Model (DCOM) provide a client/server object and interface binding architecture. Their objective is to shield client applications from server object details such as interface binding and object location by writing an object once, exposing its interface, and allowing it to be used over and over without regard to its location. The very nature of location hiding complicates application measurement. Many of the vendors of Object Request Broker (ORB) software claim to eliminate the need for application measurement because the ORBs have already been instrumented. Unfortunately, this provides instrumentation at the server (S1) and does not account for the client portion of the response time (A-I and J-B). In addition there are other unique issues involved with measuring object environments. A single client can invoke the same function (object) multiple times and the request will be serviced by different servers each time. Transaction routing may differ based on data sensitive business rules unknown to the client or even the initial server.

The application topology shown in *Figure 1* is not sufficiently complex to represent a reasonable CORBA application because the client component can actually invoke services from multiple servers, such as S2 and S3, directly. This complexity in routing make Wire Sniffing almost impossible. Even if all the required network segments could be adequately instrumented, the effort to correlate the resulting data would be massive. Benchmarking suffers from a similar complication of complexity. Because there are no client controls to select how to route messages or which servers should be used for a given function, the benchmark scripts are unable to ensure that all components of the application are measured. The issue of function reuse makes Application Instrumentation much more complex because the organization wanting the measurement may very likely not have access to the source code to make the modifications. Client Instrumentation tools may not have the visibility into the client component to distinguish between different instances of the same dialog used to drive different business transactions.

4.6 Summary

Unfortunately, the selection of which measurement technique to use is not made easy by identifying the type of application. A complex web application is not better measured one way or another. The type of information required drives the selection of the technique. For example, the need for client component information requires the use of a client component measurement technique and eliminates the usefulness of server-side measurement technique, regardless of how robust it is. Once the measurement objective has been identified, the appropriate technique, or combination of techniques, can be implemented.

5. Conclusion

The traditional view of application measurement is evolving because of the desire to understand the impact that application response time has on the overall business. Applications designed to exploit a client/server architecture greatly increase the complexity of both the computer system configurations and the applications themselves. Measurement of these applications is very complex and must focus on the business result to avoid massive data collection and analysis problems.

Where the application is measured is more important than how the measurement is implemented. This top down approach starts with the business need to determine what type of information is required and then implements the measurement technique that both provides that type of information and fits with the application. If the business need is to understand the number of call center operators to hire, then the end-to-end response time including user think time may be the best metric to use. If the business need is to understand the productivity of each of the call center operators, then the end-to-end response time excluding user think time may be a better metric. If the business need is to understand the server capacity required to meet the planned call volume, then a combination of end-to-end response time and IT transaction response time may be required. In each case, once the business need has been identified it becomes much easier to determine where, and then how, to measure the application.

The four application measurement techniques presented (Application Instrumentation, Client Instrumentation, Wire Sniffing and Benchmarking) differ in how they are implemented and their ability to collect data at a given measurement point. Other factors, such as accessibility to source code, the client operating environment (i.e., Windows), or the cost of implementation, are significant in the selection of the technique to use. The real problem arises when these factors force the selection of a technique which, for the given application, cannot be implemented at the measurement points needed for business decisions. Knowing the limitations at the beginning of the measurement effort allows the selected technique to be adjusted to meet the business needs. If such adjustment is not possible, then the scope of the effort can at least be reduced by eliminating some of the instrumentation implementation and data collection that does not support the business requirements. When thinking of application measurement, remember the axioms: “Just because we should, doesn’t mean we can” and “Just because we can, doesn’t mean we should.” Let the business need be the force that drives both the desire and the implementation of any application measurement effort.

6. References

Knight, Alan and Jonathon Haworth. 1996. Self Instrumenation - A Discussion of Requirements and Approaches. In UKCMG, Proceedings: Computer Measurement Group.

Lipovich, G. Jay. 1997. Fixing Capacity Planning’s Achilles Heel: An Approach to Managing Forecast Inaccuracy. In Computer Measurement Group, Proceedings.

McBride, Doug. 1997. Performance Management of the Desktop Client Fact or Fantasy? In Computer Measurement Group, Proceedings.

Norton, Tim R. 1998. Don’t Predict Applications When You Should Model the Business. In Computer Measurement Group, Proceedings:922-933. Anaheim, CA: CMG, Inc.

Ramanathan, Srinivas and Edward H. Perry. 1999. The Value of a Systematic Approach to Measurement and Analysis: An ISP Case Study. In SIGMETRICS, Proceedings V24 N1:232-233. Atlanta, Georgia: ACM.

Smead, Steven. 1998. Service Level Instrumentation 101 - An in depth look at how to instrument end user transactions. In Computer Measurement Group, Proceedings.

Smith, Connie U. and Lloyd G. Williams. 1998. Performance Engineering Models of CORBA-based Distributed-Object Systems. In Computer Measurement Group, Proceedings.

Thompson, George I., Javier Muñoz, and James K. DeBruhl. 1997. The Availability & Quality of SAP R/3 Workload Data For Performance / Capacity Management Process Requirements. In Computer Measurement Group, Proceedings.

Tsykin, Mike and Christopher D. Langshaw. 1999. End-to-End Response Time And Beyond: Direct Measurement of Service Levels. Computer Measurement Group Transactions (95): 41-48.