Measuring End-to-End Availability: How to Get Started

(PRESENTATION WHITE PAPER)

January 2000
Mike Bonett
Systems Management Technical Support
IBM Corporation, Advanced Technical Support
Gaithersburg, MD
bonett@us.ibm.com

Preface

Questions or comments about this publication should be sent via the Internet to **bonett@us.ibm.com**.

Acknowledgments

Many thanks to the following individuals who, over the last several years, have provided valuable experience, information, feedback and sanity checks related to the contents of this document:

- John Bishop
- Bob Campenni
- Bob Gelinas
- Pete Gordon
- Randy Greene
- Rich Grimaldi
- Eric Klein
- Kevin Miller
- Mark Nixon
- Doug Orlando
- Dave Petersen

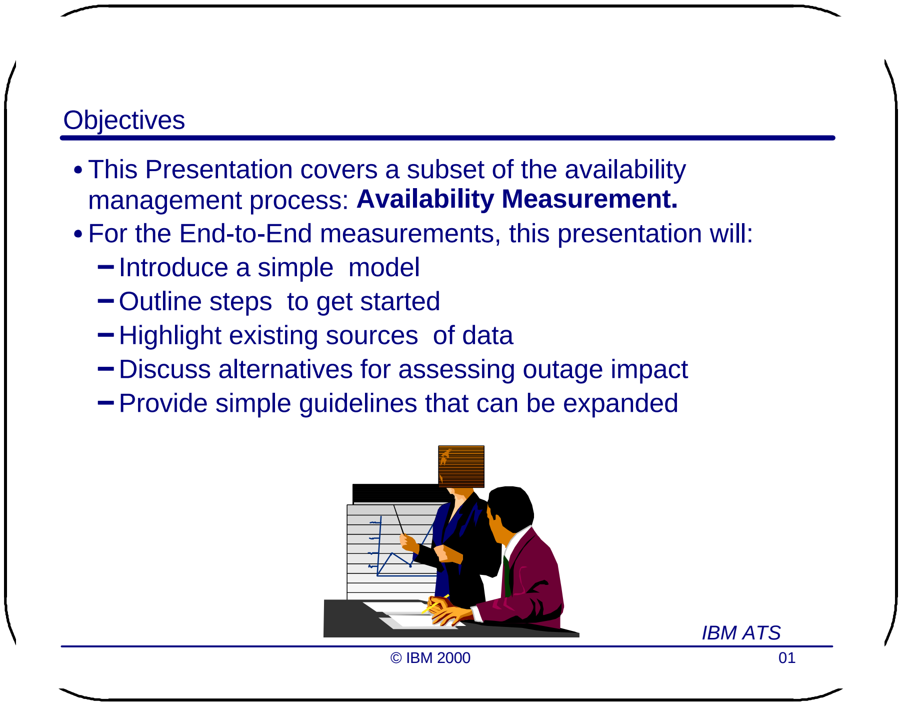
Introduction

Traditional availability measurements have focused on component availability - for example, CPU/networking hardware or a single address space. With the increasing distribution of functions and data for critical applications, the need to understand and measure availability from the end users perspective - also known as end-to-end availability - is required to provide the proper levels of service. This document outlines a simple model and some steps that can be taken to begin measuring end-to-end availability. The model and steps can be applied to any operating system and network environment. The information covered will include how to identify what should be measured, data sources and techniques that can provide measurement data, and ways to determine the impact of outages from a user/business point of view.

This paper should be used with the "Measuring End-to-End Availability: How To Get Started" presentation that the author has given at a number of IBM customer seminars. It refers to the charts and figures that are contained in the presentation.

A subset of this information was initially published in the "Measuring End-to-End Availability" article in the November 1994 issue of **Enterprise Systems Journal** magazine.

Page 01



Objectives

- This Presentation covers a subset of the availability management process: **Availability Measurement.**
- For the End-to-End measurements, this presentation will:
  - Introduce a simple model
  - Outline steps to get started
  - Highlight existing sources of data
  - Discuss alternatives for assessing outage impact
  - Provide simple guidelines that can be expanded

IBM ATS

© IBM 2000 01

Availability Management is a "rubber-meets-the-road" process within the overall management of the Information Technology (I/T) environment. If users are unable to use the applications that are supported by I/T resources, the business processes supported by the applications will suffer. Or, to put it in a positive light, the more available applications are to users, the more the business processes can be used, with corresponding benefits.

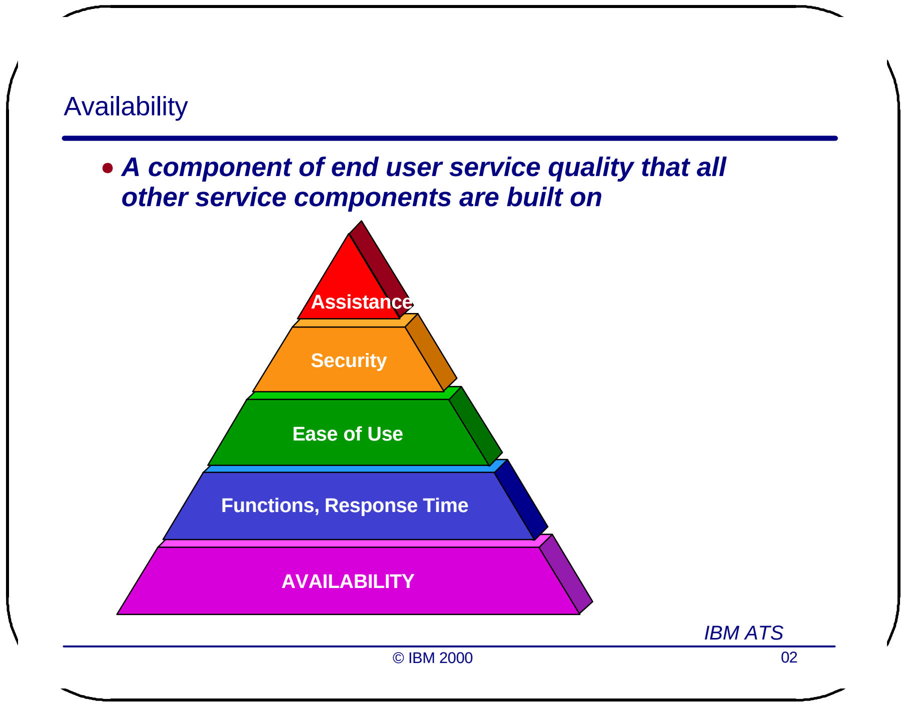
This document discusses availability measurement from the "end-user" or end-to-end perspective. It will propose a series of steps to take to help identify what should be measured, how to collect the data, and derive end-to-end measurements from the collected data. There are several actions to help this process that will be also be covered:

- The use of an "application model" to identify key points of measurement
- What data already exists, or can be created, to provide input to the measurement process
- Measurement of availability beyond "percentage available" into more business impact and cost oriented metrics.

Availability measurement is a subset of the availability management process. Without using other parts of the process - which include establishing an availability plan and taking actions to improve availability - the measurements taken cannot be effectively used.

These steps will provide simple guidelines to get started. Many times excessive energy is expended (and wasted) just on beginning such an activity; this usually occurs because a wide ranging, complex view is taken, and an attempt is made to account for all the complexities at the beginning. These guidelines, in contrast, present a way to start at a relatively simple level, but with the flexibility to expand to the desired level of complexity.

Page 02



Availability

- *A component of end user service quality that all other service components are built on*

Assistance
Security
Ease of Use
Functions, Response Time
AVAILABILITY

IBM ATS

© IBM 2000 02

A level set is needed to show the importance of availability.

Availability has been something that is taken for granted when it exists, but which everyone becomes acutely aware of when it does not exist. A basic definition of "availability" is the requirement that the information/technology infrastructure that supports business applications be in a state that allows the applications to be used by the appropriate users/departments/organizations, and that the applications can access to the data required to support the appropriate business process or processes.

Availability management" must be able to quantify availability to determine what, if any adjustments are needed. The only way to do this is through measurement. As the saying goings, "you cannot manage what you cannot measure".

An application critical to the business, for which availability is a concern, must have service levels established to define the required level of availability. Measurements are needed to validate the service levels - to determine if they are being achieved or being missed. The measurements will also identify periods of unavailability, from which the costs of these outages can be derived. This is necessary to decide the proper investment level needed to address outage situations

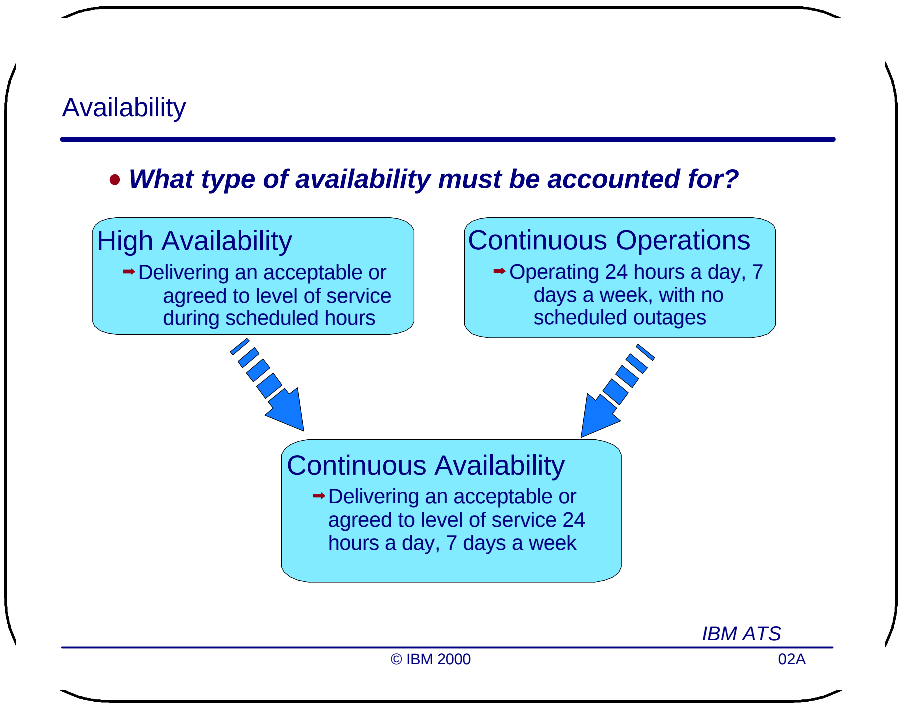
Applications are what users use to get their work done in a productive manner. Applications can also bring in revenue to the company. Application service quality is important, since higher levels of service can result in higher user satisfaction, increased user productivity, and/or increased revenue.

Application service quality is composed of several criteria. They include:

- Availability, as has been discussed on the previous page.
- Application functions to accept and process the user requests
- Response time when using the application functions, or returning requests
- Ease of using the application, including the ability to quickly learn its functions
- Assistance provided when help is needed
- Security of information access

For all service quality criteria, availability is the foundation. Without availability it is impossible to even judge or measure the quality of the other criteria. Without availability many of these criteria cease to exist. Therefore, attention need to be paid to availability.

PAGE 02A


Availability

- *What type of availability must be accounted for?*

High Availability
Delivering an acceptable or agreed to level of service during scheduled hours
Continuous Operations
Operating 24 hours a day, 7 days a week, with no scheduled outages
Continuous Availability
Delivering an acceptable or agreed to level of service 24 hours a day, 7 days a week
IBM ATS
© IBM 2000
02A

One must know the type of availability that will be measured. It will influence the type of data that is collected, and the type of analysis done of the data to provide meaningful information. The three types to be aware of are:

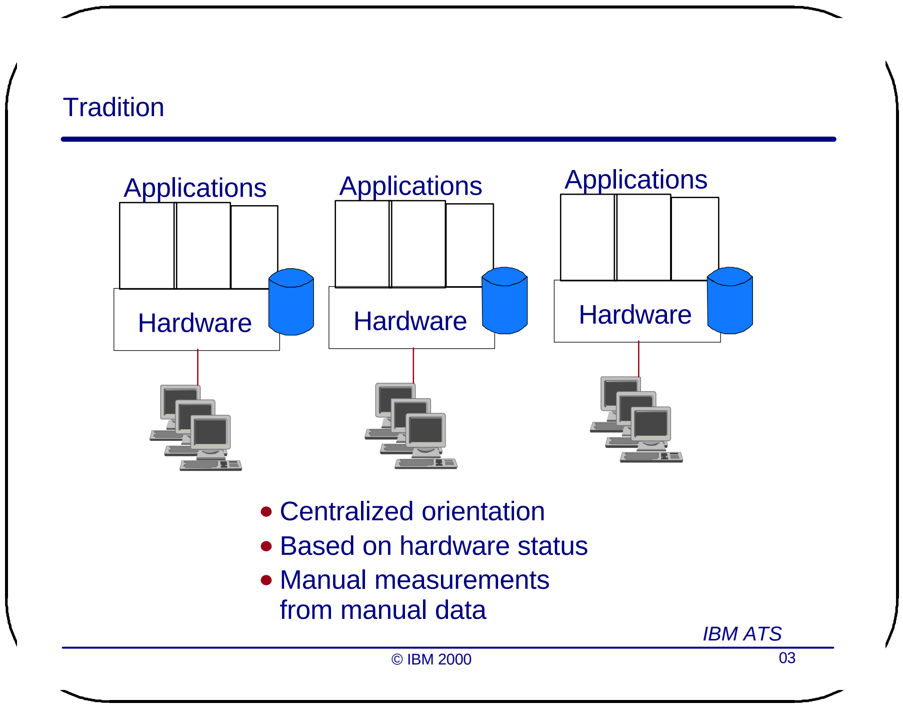
1. **High Availability** means **an acceptable or agreed to level of end user service during scheduled periods**. To provide high availability an acceptable or agreed to objective between the providers of service and the end users is needed. This is usually included in a documented **service level agreement**.

2. **Continuous Operations** means that the user (human or workload) has access at any time. However, during periods of this access not all functions may be available. For example, certain data may only be in read-only access, or certain software functions may be disabled. There may or may not be service level criteria associated with a continuous operations environment. If there is, it is usually limited to significant interruptions, such as the number of IPLs permitted during a time period.

3. **Continuous Availability** combines the best of both worlds from Continuous Operations and High Availability. This is sometime written as the formula **CA = HA + CO**. It means that acceptable or agreed to service, as documented in a service level agreement, is being provided to users at all times. Continuous availability is a **true** '24 by 7" environment.

Measurements will have to take into account the availability type to provide accurate information. There is no reason to provide continuous availability measurements when all that matters are the high availability periods (unless there are plans to move to a continuous availability environment). Likewise, there is no reason to take complex availability measurements when all that matter are continuous operations. The important point here is to understand what type of availability needs to be measured before the measurements are implemented..

PAGE 03


Tradition

Applications
Applications
Applications
Hardware
Hardware
Hardware

- Centralized orientation
- Based on hardware status
- Manual measurements from manual data

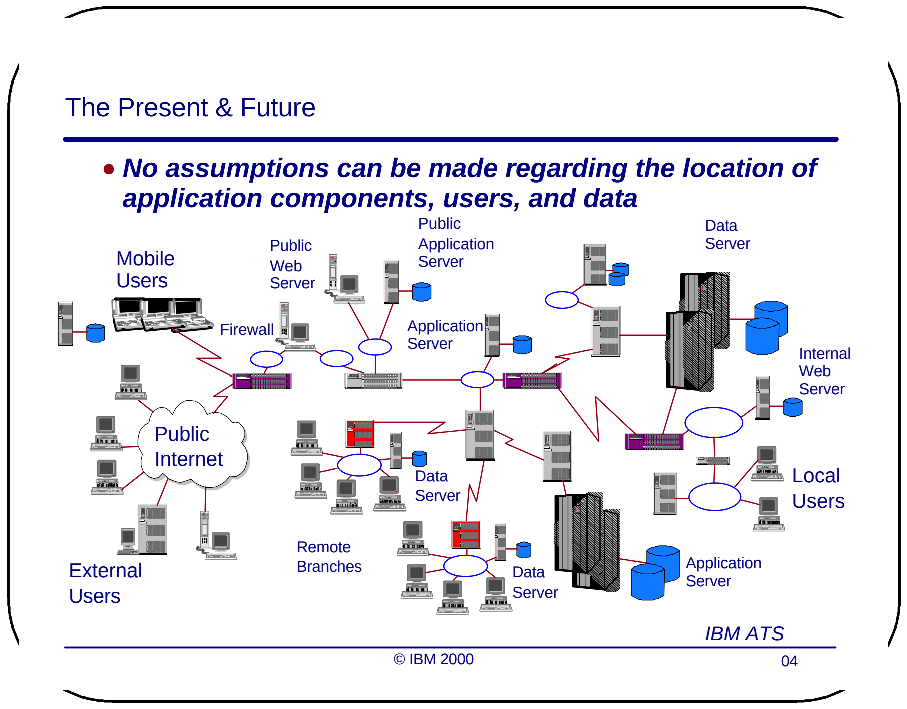
IBM ATS

© IBM 2000 03

Availability measurements were traditionally based on the state of the hardware platform where the application code was located. The simplest measurement was based on time between initial program loads (IPLs). The fewer IPLs that were done, the better the availability measurement; for example, it was assumed that, if no IPLs were encountered in a given month, 100% availability had been achieved. Over time, additional measurements based on the state (up or down) of application supporting address spaces and subsystems in the mainframe were also used to determine application availability. These measurements also provided a very centralized view.

The measurement process was carried out in a manual fashion. Someone, or an organization, would review information from problem tickets or operator logs (documents in themselves created manually) and develop reports showing what percentage of availability particular components or "applications" achieved. In some cases "availability measurement" applications were written where one could input the information online into a database and report would be generated from the data.

PAGE 04



The Present & Future

- ***No assumptions can be made regarding the location of application components, users, and data***

IBM ATS

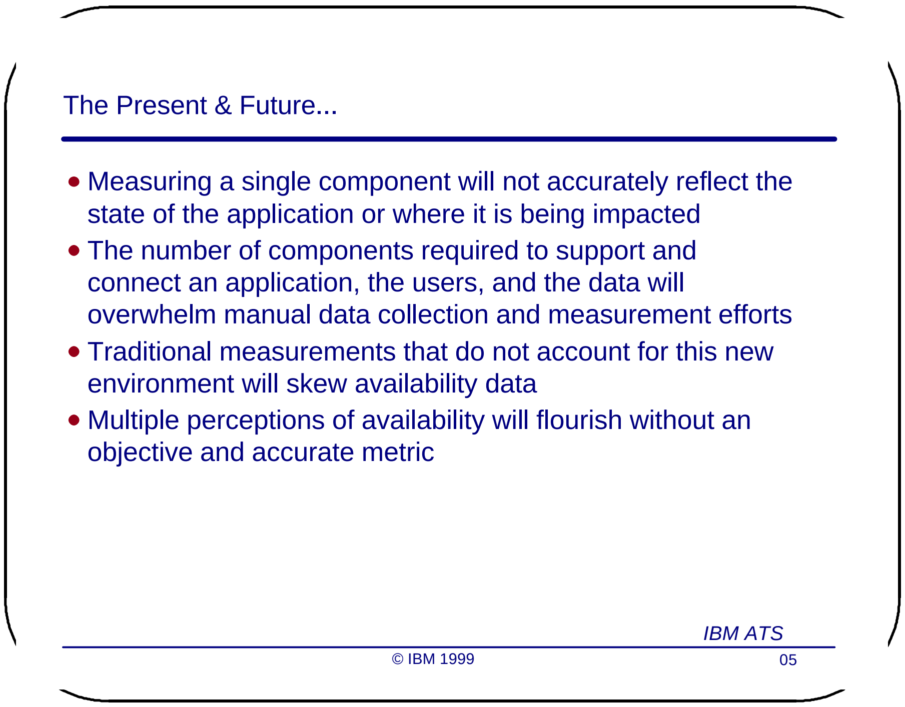
© IBM 2000

04

When environments were very centralized and had simple networks, traditional measurements gave an accurate reflection of the real availability that users experienced. However, the application infrastructure environment has radically changed, due to the onset of client/server computing and network-centric computing. Additional factors new exist which the measurements have to consider:

- Users, application, and data can be located anywhere in the enterprise. No longer does the "everything in one location" hold true, as application functions increase, and a broader range of users require access to those functions.
- Because of this location flexibility, networking protocols and hardware components are needed to provide a foundation for communication among users, application, and data.
- "Middleware" protocols (functions that exchange information between multiple locations connected via a network) are required to distribute functions and data where they are needed the most.
- The applications being placed in this environment are very critical to the business processes they support. There may be little or no ability to "fall back" to a manual process.
- Application outages in this environment are now visible beyond the I/T support organization, or the affected internal users. Other internal users, external users, customers, and even the press may discover (and publicize) unavailability.

PAGE 05



The Present & Future...

- Measuring a single component will not accurately reflect the state of the application or where it is being impacted
- The number of components required to support and connect an application, the users, and the data will overwhelm manual data collection and measurement efforts
- Traditional measurements that do not account for this new environment will skew availability data
- Multiple perceptions of availability will flourish without an objective and accurate metric

IBM ATS

© IBM 1999 05

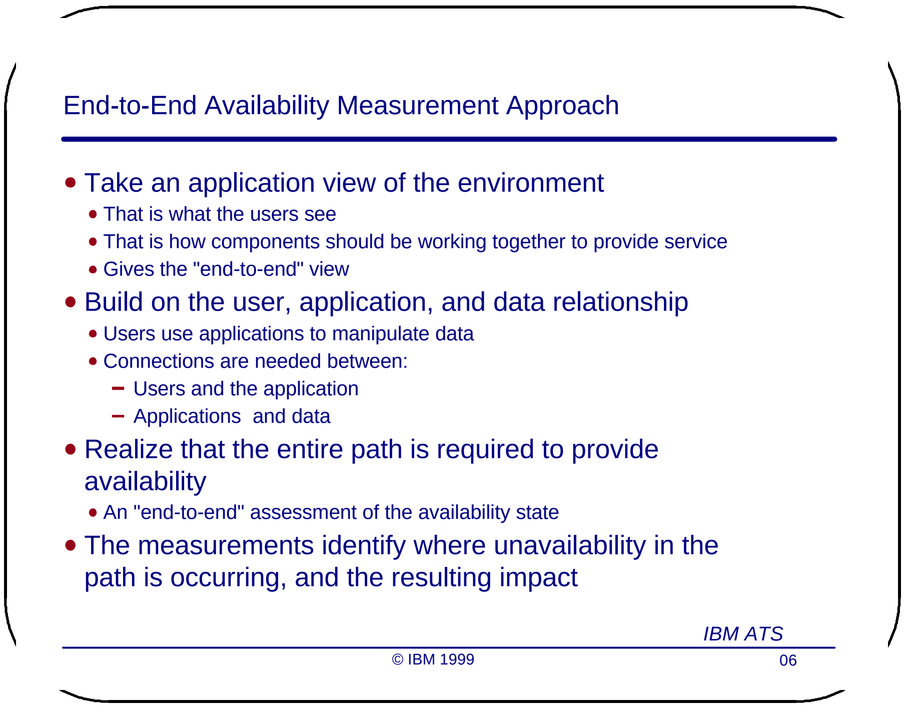
Because of these factors, continuing to take and use the availability measurements in the same manner will lead to problems:

- The state of a single component will not reflect the availability seen by the end user. The component may be active, but other components that support the application, users, data, and connections may be causing unavailability to users or a group of users.

- The sheer number of components may overwhelm attempts to manually track availability. A simple application structure that includes an end user workstation, application logic on both the workstation and the server, and data distributed among two servers, can require sixty or more components (both hardware and software) to provide availability; some number of these will have to be tracked.

Measurements that do not account for these factors will be reporting inaccurate, and perhaps higher, availability than is actually being experienced. This can lead to different perceptions of availability from the central location and from the users. The central location may see high availability, but the users are seeing lower availability because the proper components are not being measured. As in the story of two people looking at different sides of the same shield, and one sees gold, the other sees brass, the users and the central location both have different - and accurate - views of the situation.

Page 06



End-to-End Availability Measurement Approach

- Take an application view of the environment
  - That is what the users see
  - That is how components should be working together to provide service
  - Gives the "end-to-end" view
- Build on the user, application, and data relationship
  - Users use applications to manipulate data
  - Connections are needed between:
    - Users and the application
    - Applications and data
- Realize that the entire path is required to provide availability
  - An "end-to-end" assessment of the availability state
- The measurements identify where unavailability in the path is occurring, and the resulting impact

IBM ATS

© IBM 1999 06

The solution to this problem is to measure "end-to-end" availability - availability based on all the critical components an application requires for its proper execution. This is the availability the users will see, and the availability they will use to judge the service level that information systems technology is providing them. The approach for end-to-end measurements build on the following foundation:

The following assumptions will be used to define the scope of this "end-to-end" activity:

w Take an application view of the environment. Availability is often measured from a pure component view, with little or no understanding of what the measurement means from an application perspective. The application view first determines how the application operates, then uses the user-application-data assumption described earlier to identify key components and connections that are required for proper execution.

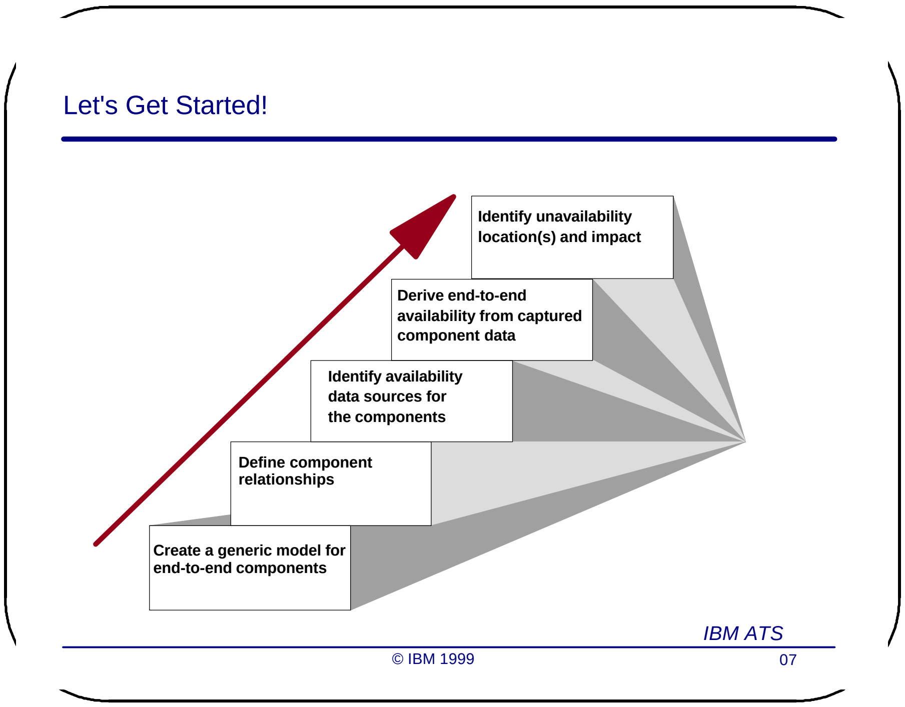
w Build on the user, application, and data relationship, which is very simple to understand:

  x Users use applications to manipulate data. Every critical I/T application can be described in this manner.

  x Connections must exist between the users and the application, and the application and the data. The connection from the users to the data is through the application.

- Components that support those areas - the users, applications, data, and connections - must be in an available state to achieve end-to-end availability. Multiple components must be identified to determine the overall availability.
- When unavailability does occur in this path, the location of where it occurs is necessary to determine its scope, in terms of affected users and/or workload. This contributes to identifying what improvements are needed and to what degree they should be implemented.

In view of this approach, end-to-end availability must be **measured**, as objectively as possible. This measurement is used not only to determine "what is really being achieved", but also to determine "how much improvement is possible and where are changes needed?".



Let's Get Started!

Identify unavailability location(s) and impact
Derive end-to-end availability from captured component data
Identify availability data sources for the components
Define component relationships
Create a generic model for end-to-end components

IBM ATS

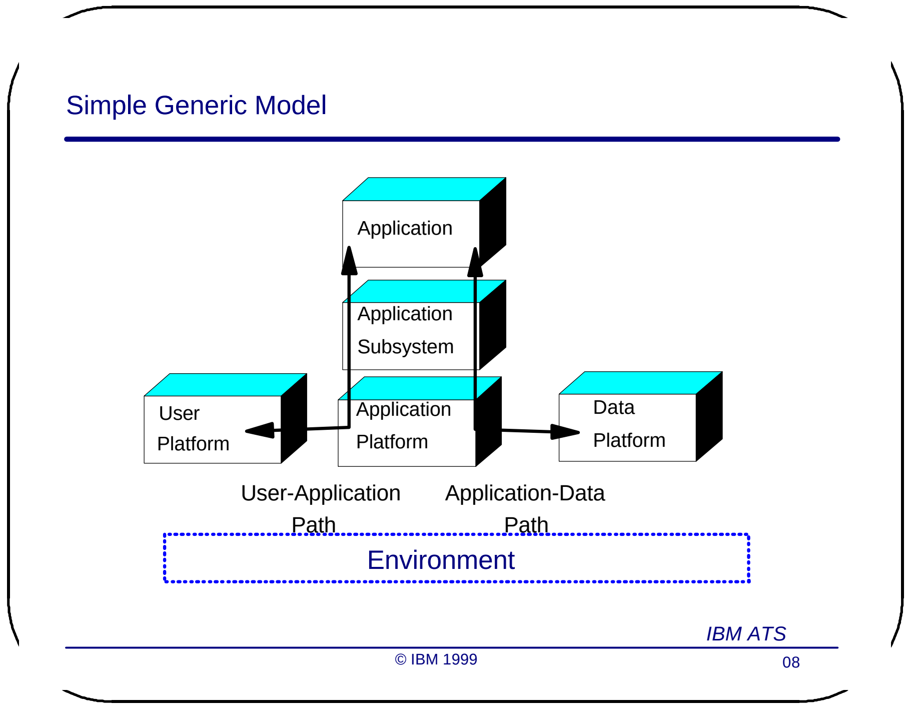
© IBM 1999 07

The rest of this document describes the steps to model and begin deriving a true picture of end-to-end availability and its business impact when unavailability occurs. These steps use the assumptions and apply the characteristics identified on the preceding pages. They can be used in any application supporting environment. They are designed to provide a simple way to get started and, based on the requirements of a situation or environment, to be reapplied to as great a level of detail as desired. In order, the steps are:

1. Create a generic model for end-to-end components.
2. Define component relationships.
3. Identify availability data sources for the components.
4. Derive end-to-end availability from measured component data.
5. Identify unavailability locations and impact.

This is an iterative process; the findings from any step may be used to go back and refine a previous step. As the measured environment changes, the impact to the process steps will have to be looked at and modified as appropriate.

Each of these steps will now be discussed in greater detail.



The first step is to **create a generic model for the end-to-end components.** Using the assumptions and approach that has been described, identifying the components that support and connect end users, applications and data is required. A model helps simplify the identification process and are the starting point for this activity.

The choice of "which model to use" is installation dependent. Variables such as the type of application, the level of detail, and the scope across the infrastructure are all factors. A useful set of models is found in the **IBM Systems Journal**, Volume 32, Issue 4 (1993) article called "Application Reference Designs for Distributed Systems".

The model pictured on this chart is a simple, generic model that will be used in this document. Any component can be mapped into one or more of the seven categories shown. These are "logical" categories that may include one or more physical components. Components in all seven areas must be available to provide end-to-end availability.

1. The **User Platform** is where the end user access to the application begins, and/or where application results are delivered. The user platform components can support any of the following:
   - A real person using the application.
   - A device receiving output from an application.

- A connection between applications. if application A "feeds" data into application B, the components that support this feed could be viewed as the user platform for application B.

2. The **user-application path** is the infrastructure that connects the user platform and the application platform. This can be as simple as a control unit, or as complex as LAN adapters, segments, bridges, switches, routers, communications lines, etc.

1. The **application platform** is the hardware and hardware operating system where the application code is physically located. These are grouped together since the hardware is fairly useless without the operating system, and vice versa. If the application code is physically distributed, this may mean a combination of application platforms support the application. If the platform is a "clustered" platform (such as a sysplex, High Availability Coupled Multiprocessing (HACMP) configuration, or NT cluster configuration), and the application code be moved to run anywhere on the cluster, the entire hardware and operating system cluster can be viewed as the application platform.
2. The **application subsystem** allows the resources of the application platform (CPU cycles, memory, system functions, etc.) to be used by the application code. Some types of "Middleware" can be thought of as the application subsystem. Software that provides transaction processing and/or data access and management can fall into this category. CICS, which runs on various application platforms, or the popular relational data base management software products are examples of the components that would map to this section of the model.

1. The **application** is the actual software code that processes the users work request and manipulates the data associated with the request. Applications support and/or automate business functions such as

   - Payroll
   - Customer service
   - Work request
   - Account inquiry

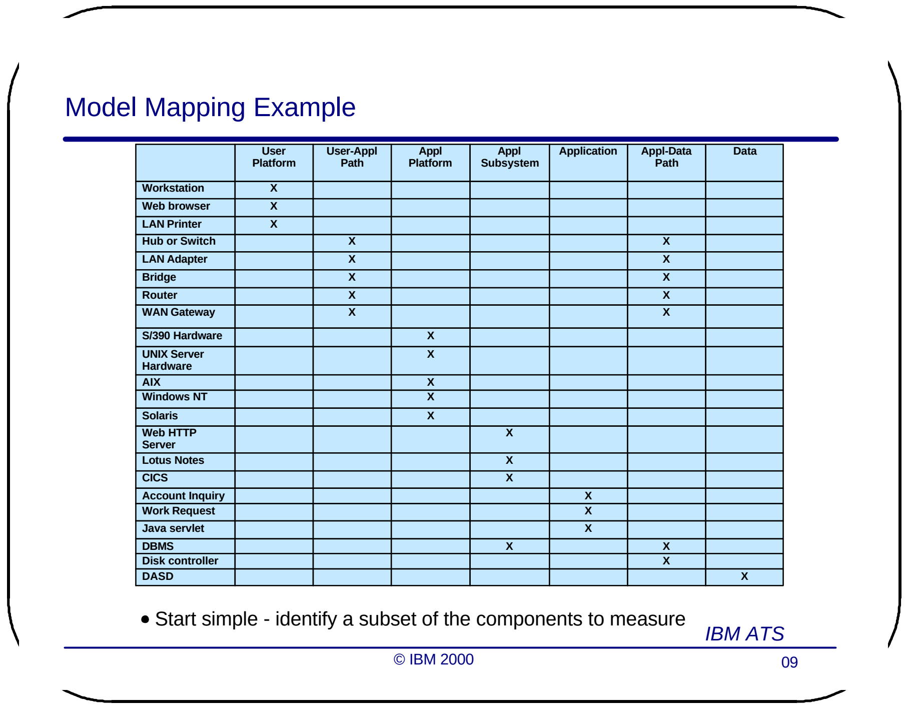
1. The **application-data path** allows the application to access the data needed to support the application functions. In some cases components in this path are same (or the same type) of components in the user-application path, depending on where the data is located and how it is distributed.

1. The **data platform** is both the physical and logical components needed for the data. The physical components are where the data resides, such as disk and tape. The logical components are used by the application platform to access the data - an access method, or data management system, for example.

Outside of the "system" components the model call also include "environment" components that are not a part of the technology infrastructure, but can influence availability. Such items as UPS, cooling facilities, etc. can be mapped to these categories by being associated with the component that they support.

This simple model can be applied to centralized, distributed, and client server environments, because there is no requirement that any of the components physically or logically reside in a specific place.

PAGE 09



Model Mapping Example

	User Platform	User-Appl Path	Appl Platform	Appl Subsystem	Application	Appl-Data Path	Data
Workstation	X						
Web browser	X						
LAN Printer	X						
Hub or Switch		X				X	
LAN Adapter		X				X	
Bridge		X				X	
Router		X				X	
WAN Gateway		X				X	
S/390 Hardware			X				
UNIX Server Hardware			X				
AIX			X				
Windows NT			X				
Solaris			X				
Web HTTP Server				X			
Lotus Notes				X			
CICS				X			
Account Inquiry					X		
Work Request					X		
Java servlet					X		
DBMS				X		X	
Disk controller						X	
DASD							X

- Start simple - identify a subset of the components to measure

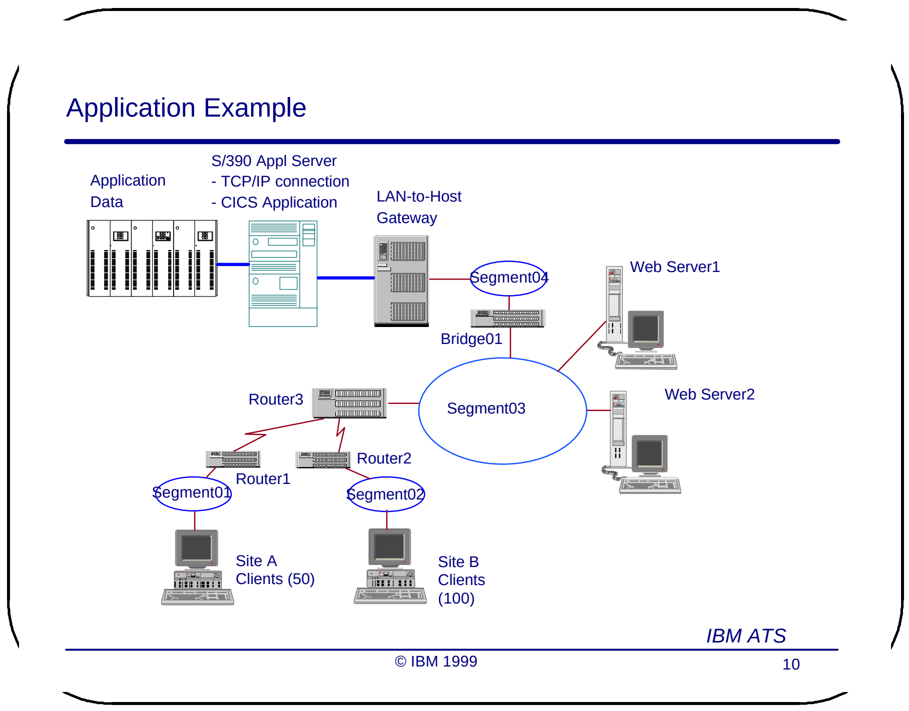
IBM ATS

© IBM 2000

09

Here is an example of how some common system and network components would be mapped to the model categories. Note that a component can be associated with one or more categories.

At this stage the important action is to identify the key components and map them to the model. It will be too complicated to map every component; a "rule of thumb" would be to start with 3-10 components that are mapped to at least the user platform, user-application path, and application platform categories. Components that are not being tracked should (hopefully) be recorded in the problem management process if problems affecting the component should occur; this will be discussed later as a means of helping identify what additional components should be tracked.



Application Example

IBM ATS

© IBM 1999

10

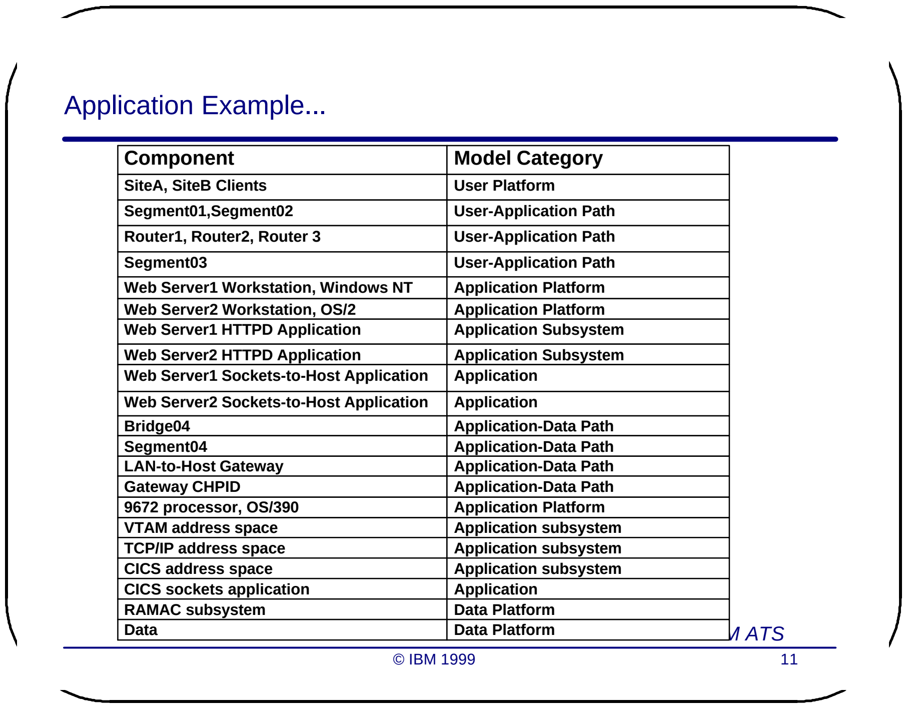
An example application will to used to show how to apply the steps for measuring availability. It uses a e the infrastructure supporting of the most common and growing applications today - the "web-enabled" or “e-business” application. These applications allows end users on the intranet or internet to access functions and information in “legacy” applications via web connections.

The application picture has uses on an intranet accessing a S/390 server. The major components of the infrastructure that support this application are:

- The end-user, or client, platform, containing the web browser, communications protocol stack, and LAN connection adapter.
- The LAN segment the clients are located on (which physically is a hub or switch).
- A bridge that connects the user LAN segment to the web server LAN segment.
- The web server LAN segment.
- The web servers with the "web" portion of this application. The user requests are initially processed on these servers. Two servers are used for both workload balancing and redundancy.
- The LAN-to-host gateway (typically a router or communications controller) that connects the web server LAN segment to a S/390 server.

- The S/390 server that contains another piece of the application code. The web server will invoke an CICS program on the S/390 server via TCP/IP that will retrieve the appropriate data based on the user query.
- The DASD array where the data physically resides.

PAGE 11



Application Example...

Component	Model Category
SiteA, SiteB Clients	User Platform
Segment01,Segment02	User-Application Path
Router1, Router2, Router 3	User-Application Path
Segment03	User-Application Path
Web Server1 Workstation, Windows NT	Application Platform
Web Server2 Workstation, OS/2	Application Platform
Web Server1 HTTPD Application	Application Subsystem
Web Server2 HTTPD Application	Application Subsystem
Web Server1 Sockets-to-Host Application	Application
Web Server2 Sockets-to-Host Application	Application
Bridge04	Application-Data Path
Segment04	Application-Data Path
LAN-to-Host Gateway	Application-Data Path
Gateway CHPID	Application-Data Path
9672 processor, OS/390	Application Platform
VTAM address space	Application subsystem
TCP/IP address space	Application subsystem
CICS address space	Application subsystem
CICS sockets application	Application
RAMAC subsystem	Data Platform
Data	Data Platform

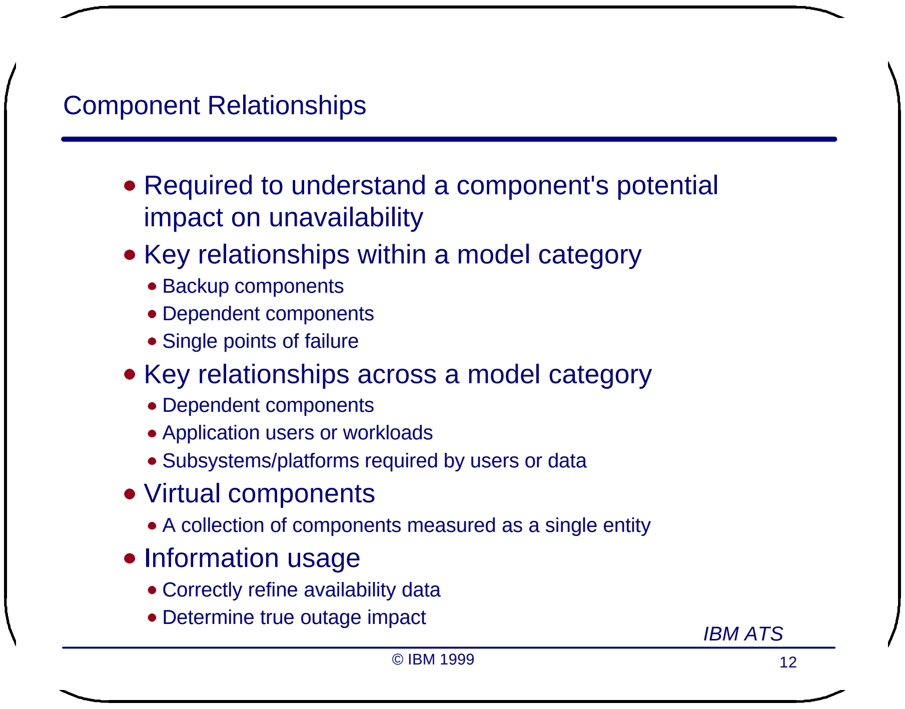
M ATS

© IBM 1999 11

This table shows how the major components of the application would be mapped to the generic model. Always remember that mapping to the model is an art, not a science. It is more important to identify the major components that to debate over which category, or set of categories, a component should be mapped to.

The level of component detail can be as high or low as desired. It will depend on what degree of component availability is desired (or is able) to measured. For example, individual CICS Terminal Owning Regions (TORs) and Application Owning Regions (AORs) could be identified as part of the application subsystem for the ATM application; or, multiple adapters into the web server could be individually identified.

Page 12



Component Relationships

- Required to understand a component's potential impact on unavailability
- Key relationships within a model category
  - Backup components
  - Dependent components
  - Single points of failure
- Key relationships across a model category
  - Dependent components
  - Application users or workloads
  - Subsystems/platforms required by users or data
- Virtual components
  - A collection of components measured as a single entity
- Information usage
  - Correctly refine availability data
  - Determine true outage impact

IBM ATS

© IBM 1999 12

Placing components into model categories will permit the second step, **define component relationships**, to completed. Understanding the relationships (sometimes called "linkages") at this point in the process is important for several reasons:

- It is needed to accurately derive end-to-end availability, and to show how each component influences that availability.
- It will reveal, for a given component, how it can impact other components, users, and the workload. This will be important when measuring to identify the impact of an outage in terms beyond "percentage of unavailability".
- It will show, of the components identified in the model mapping, the relative importance of each one, and therefore which ones may be most important to begin measuring.

Component Failure Impact Analysis (CFIA) is an exercise some environments have undertaken to identify the impact of a component failure and to implement actions to eliminate/reduce the impact of that failure. The relationship activity is based on physical and logical configuration information, which is a subset of the CFIA activity; this can be extracted from the CFIA data. However, without a CFIA it is still possible to come up with the linkage relationships that are needed for accurate availability measurement.

What are the relationships that have to be identified? For measurement purposes the following relationships must be identified:

- Components that back each other up. If component A is down, but component B can take over its function, and vice versa, then A and B are backup components to each other. This usually occurs among components within the same generic model category.
- Components that depend upon another component for operation. If components A and C cannot be active unless component B is, then both A and C are components dependent on Component B. This can occur among components in the same or different model category.
- Single points of failure. If component A fails, and there is no replacement/backup for it, then component A is a single point of failure. Single points of failures with most directly affect end-to-end availability, since they must be active to allow the end-to-end path to function.
- Where the application users and/or workloads are located. If a component fails, knowing what users are affected is useful input to determining the outage scope.
- The subsystems and platforms required for an application. Should a platform or subsystem component fail, what applications are affected is useful input to determining the outage scope.

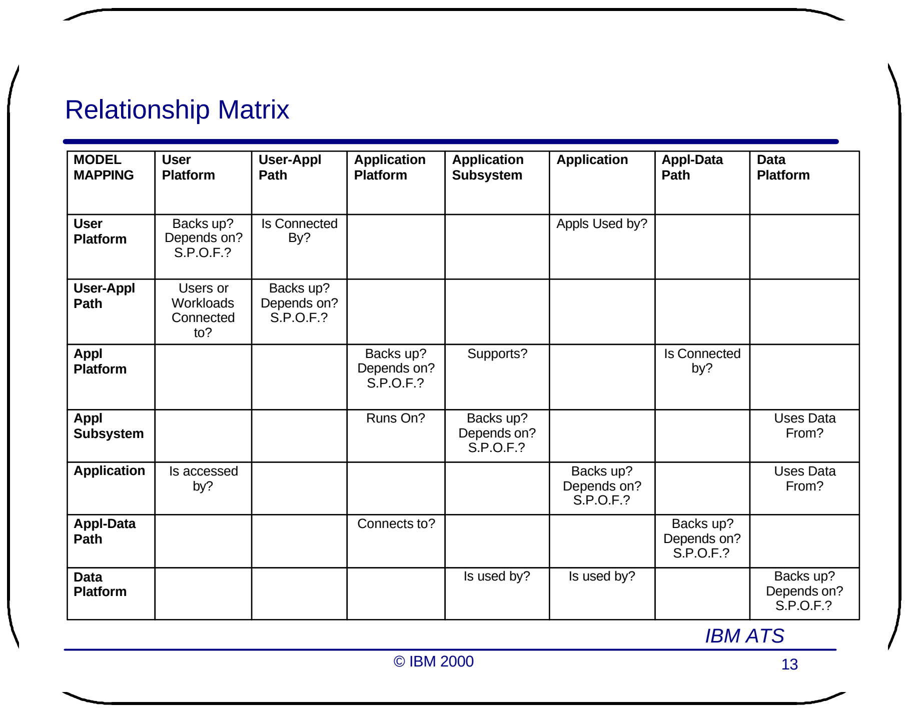
One can also choose to create, for measurement purposes, virtual components. These are collections of components all of which must be functioning for the virtual component can be working. For example, a data path between two sites may consist of two routers and the telecommunications line and equipment between them. This path can be a virtual component, made up of the two routers and the line connection. If any of the real components are down, the virtual component - the path - is down. This can help simplify the measurements.

Establishing these relationships allows the measured and collected data to be properly refined into an availability measurement. For example, suppose component A has been mapped to a category model, and incurs a 50 minute outage. Could this have impacted the end-to-end availability, and how much? It depends on whether component A:

- Is a single point of failure - it would affect the end-to-end availability
- Has a backup - it might not affect end-to-end availability if the backup component was active
- Had a set of users associated with it - if end-to-end availability was affected, the scope may have only been that set of users
- Had a workload associated with it - if end-to-end availability was affected, the impact would depend of the level of work normally being processed at the time of the outage.

The question can then be answered properly if the relationship information is known.

PAGE 13



Relationship Matrix

MODEL MAPPING	User Platform	User-Appl Path	Application Platform	Application Subsystem	Application	Appl-Data Path	Data Platform
User Platform	Backs up? Depends on? S.P.O.F.?	Is Connected By?			Appls Used by?		
User-Appl Path	Users or Workloads Connected to?	Backs up? Depends on? S.P.O.F.?					
Appl Platform			Backs up? Depends on? S.P.O.F.?	Supports?		Is Connected by?	
Appl Subsystem			Runs On?	Backs up? Depends on? S.P.O.F.?			Uses Data From?
Application	Is accessed by?				Backs up? Depends on? S.P.O.F.?		Uses Data From?
Appl-Data Path			Connects to?			Backs up? Depends on? S.P.O.F.?	
Data Platform				Is used by?	Is used by?		Backs up? Depends on? S.P.O.F.?

IBM ATS

© IBM 2000 13

This matrix uses the generic model categories to show, depending on where a component is mapped, the type of relationship information needed.

The rows and columns are the model categories, Each intersection contains examples of questions to determine if there is a relationship with a component in the row category with a component in the column category. For example, for the "user platform" row and the "user-application path" column, the entry is interpreted as "which components in the user platform are connected to which components in the user-application path?". This can be answered from a physical configuration diagram. For the "user platform" row and the "application platform" column, the question "what user platforms use which applications?" can be answered from application or operational documentation.

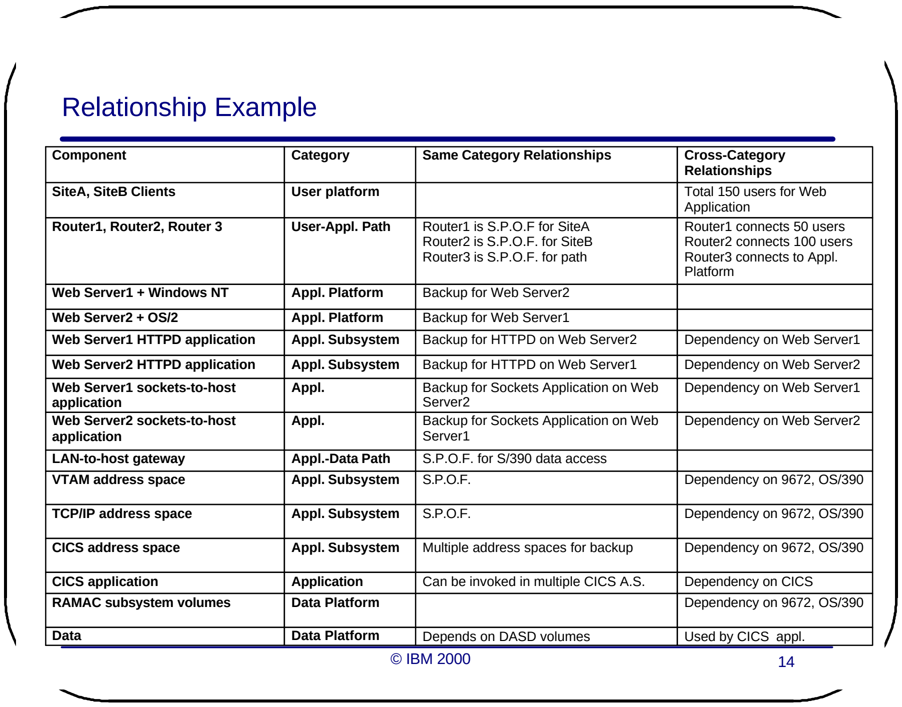
The diagonal sections of the matrix compare components within the same category. Note that the three primary relationships among these components to be determined are:
1. Backups/redundancies
2. Dependencies
3. Single points of failure

This matrix does not show all the possible relationships that exist. It highlights a small number that provide a good starting point. These relationships can also be used to derive other relationships:

- Application unavailability impacts users. So, users have to be associated with the application they require. This can be on a department or geographic basis.

- Application Subsystem unavailability impacts applications. One can then derive the user impact of an application subsystem outage if the linkages between applications and application subsystems have been identified.

PAGE 14



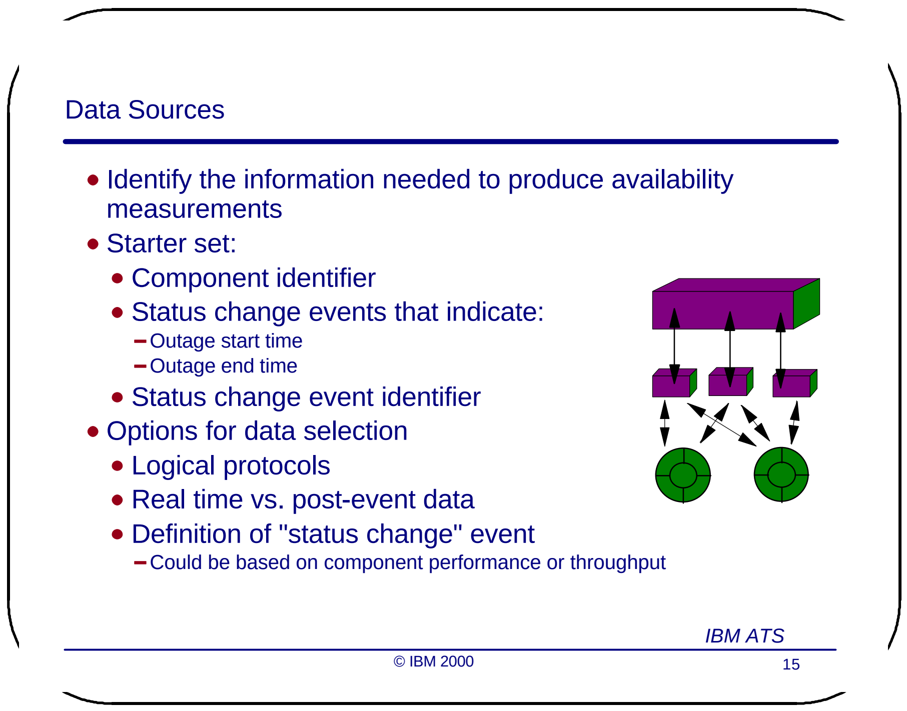
Relationship Example

Component	Category	Same Category Relationships	Cross-Category Relationships
SiteA, SiteB Clients	**User platform**		Total 150 users for Web Application
Router1, Router2, Router 3	**User-Appl. Path**	Router1 is S.P.O.F for SiteA Router2 is S.P.O.F. for SiteB Router3 is S.P.O.F. for path	Router1 connects 50 users Router2 connects 100 users Router3 connects to Appl. Platform
Web Server1 + Windows NT	**Appl. Platform**	Backup for Web Server2	
Web Server2 + OS/2	**Appl. Platform**	Backup for Web Server1	
Web Server1 HTTPD application	**Appl. Subsystem**	Backup for HTTPD on Web Server2	Dependency on Web Server1
Web Server2 HTTPD application	**Appl. Subsystem**	Backup for HTTPD on Web Server1	Dependency on Web Server2
Web Server1 sockets-to-host application	**Appl.**	Backup for Sockets Application on Web Server2	Dependency on Web Server1
Web Server2 sockets-to-host application	**Appl.**	Backup for Sockets Application on Web Server1	Dependency on Web Server2
LAN-to-host gateway	**Appl.-Data Path**	S.P.O.F. for S/390 data access	
VTAM address space	**Appl. Subsystem**	S.P.O.F.	Dependency on 9672, OS/390
TCP/IP address space	**Appl. Subsystem**	S.P.O.F.	Dependency on 9672, OS/390
CICS address space	**Appl. Subsystem**	Multiple address spaces for backup	Dependency on 9672, OS/390
CICS application	**Application**	Can be invoked in multiple CICS A.S.	Dependency on CICS
RAMAC subsystem volumes	**Data Platform**		Dependency on 9672, OS/390
Data	**Data Platform**	Depends on DASD volumes	Used by CICS appl.

© IBM 2000 14

Applying this step to our example application results in this table. Some of the key findings:

- Several components (the routers, LAN-to-host gateway, S/390 communications software) are single points of failures. These are prime candidates for measurement.
- The web servers and associated application subsystems and applications back each other up, so the application is available as long as one of them is active (and of course all other components are active)
- The application platforms (web servers and S/390 server) have several components dependent on them. If the collected data shows the application platforms are down, it also means that these dependent components will be down.



Data Sources

- Identify the information needed to produce availability measurements
- Starter set:
  - Component identifier
  - Status change events that indicate:
    - Outage start time
    - Outage end time
  - Status change event identifier
- Options for data selection
  - Logical protocols
  - Real time vs. post-event data
  - Definition of "status change" event
    - Could be based on component performance or throughput

IBM ATS

© IBM 2000 15

The third step is to **identify availability data sources for the components**.

After the important components that need to (or are desired to) be measured to derive end-to-end availability are selected, the critical activities to measure them are:

- Determining what data has to be collected
- Finding the data
- Collecting the data

These topics will be covered here at an overview level. The white paper **Finding and Collecting Availability Measurement Data** (available from the author**)** covers, in much greater detail, specific data sources, collection methods and techniques, and available products and tools.

For each component measured a relatively small amount of information is sufficient to start with. The data collected should contain:

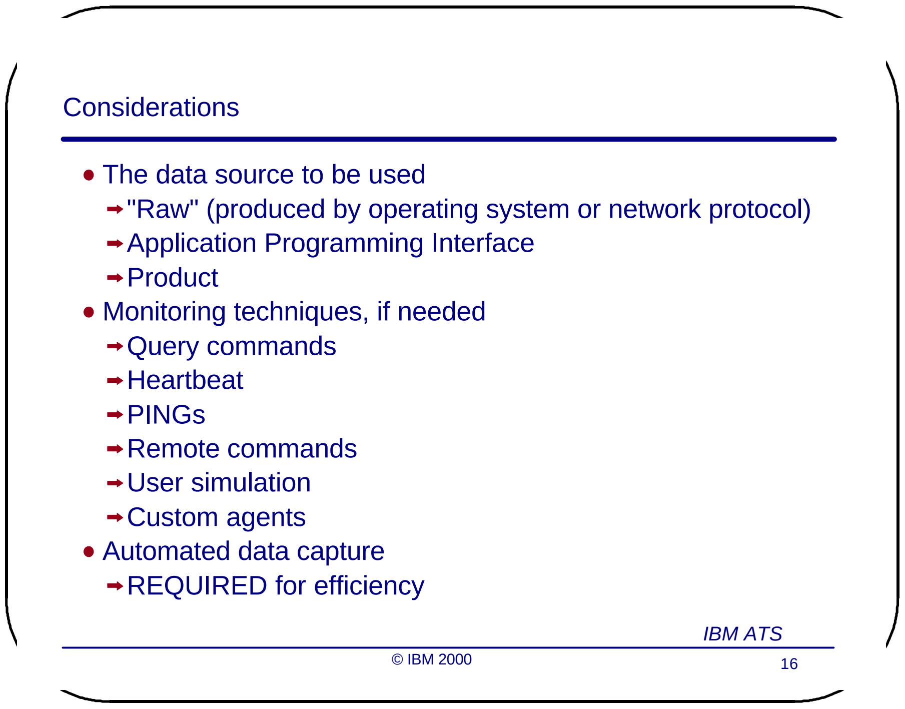
- Component identifier. Information that uniquely identifies the component being measured.
- Status change events. They contain information that includes the new status (“up” or “down”, which will be discussed later) and the date and time the status change occurred.

- Status change event identifier. This is the event (or event source) indicating the status change (event ID, monitor name, etc.).

This is a small amount of information, and multiple options exist for obtaining it from a given component. If the component participates in multiple system and network protocols, each protocol can provide this information. There also may be both real time (event obtained as it occurs) and log/audit (event recorded and scanned for at a later time) data sources with this information. There is no "right or wrong" source when multiple sources could be used; it depends on the protocols the application uses and how quickly the installation desires to capture this information.

The definition of what constitutes an "outage start" and "outage end" for a component can vary. Did the outage actually start when the component became inactive? Or did it start when the component was active but unable to provide service at a particular rate (and in fact never became inactive)? The installation can best decide. based on experience, if a particular performance threshold is the real indicator of a component outage as related to the application.

PAGE 16



Considerations

- The data source to be used
  - "Raw" (produced by operating system or network protocol)
  - Application Programming Interface
  - Product
- Monitoring techniques, if needed
  - Query commands
  - Heartbeat
  - PINGs
  - Remote commands
  - User simulation
  - Custom agents
- Automated data capture
  - REQUIRED for efficiency

IBM ATS

© IBM 2000

16

The range of existing component technologies requires integrating data from multiple sources to get an end-to-end picture. This will take some work to do but is not an impossible task. Sources for the data vary, and some will be better suited than others. Categories to consider include:

w Data directly produced by operating systems or network protocol. This is “raw” data in the sense that additional work is needed to extract the measurement information. The data may come from a structured protocol (SNA Alert, TCP/IP trap, OSI/CS alarm, etc.), or an unstructured protocol (messages, text display, etc.). Examples are:

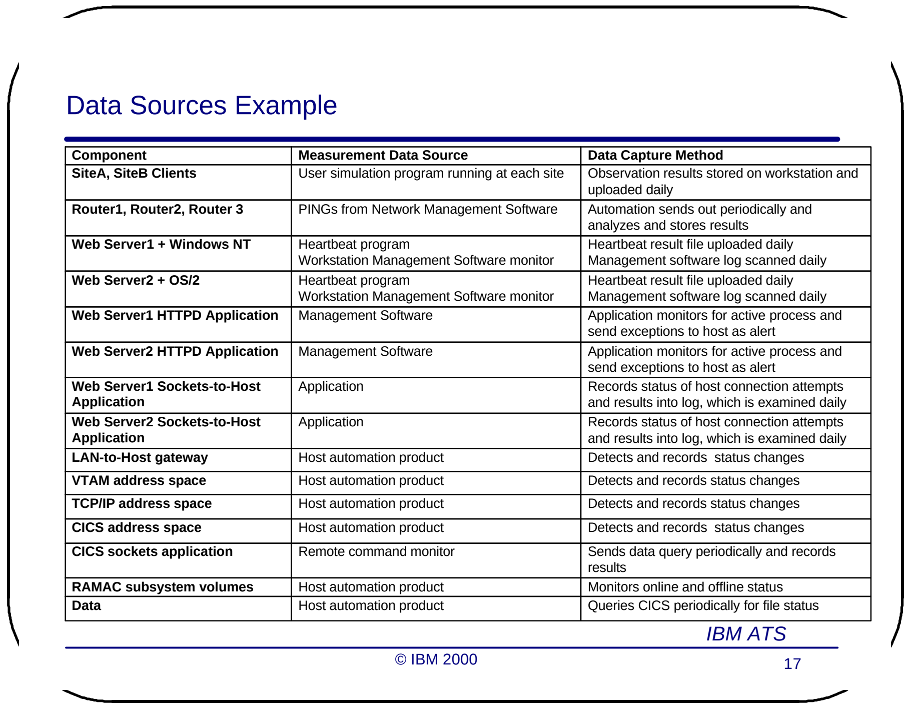
  x SMF records, for OS/390 and MVS platform components
  x S/390 messages, for OS/390 and MVS platform and attached components
  x VTAM messages, for components participating in an SNA network
  x SNA alerts, for components will the ability to produce alerts
  x SNMP traps, for components in TCP/IP networks
  x AS/400 message queues, for AS/400 platform components
  x UNIX console log and error log, for UNIX platforms
  x Windows NT event and application logs, for Windows NT platform and attached components
  x Netware Server console log, for Netware components

- Application Program Interface (API). The component, or the platform the component runs on, may provide an API that can be used to obtain specific status information.
- Products (generic or component-specific) that allow access to their data from other products. These provide information on the specific types of components they are designed to monitor. These are products with event and/or performance monitoring capabilities (many systems management and performance monitoring products provide these functions).

When these sources do not provide the desired data, active monitoring techniques can be used to query and create component status data for use in measuring availability. The query can be written using platform functions are can be provided by a product. These techniques include:

- Query commands issued by automation to determine component status
- Heartbeat functions to record status at regular intervals
- PINGs to query and receive responses from components
- Remote commands to send a command to another component and receive a command response
- User simulation to issue actual application transactions and record the result
- Custom agents provided by products that include measurement functions

Automation products play an important role in this step. They can capture data from the “raw” sources, or invoke the monitoring techniques and capture the results. They can screen and filter the data to extract the needed information for the required components. Automation products have interfaces to a wide range of components and protocols, and can integrate and consolidate information from different sources for consistency. Even producing the desired reports can be controlled by automation. For end-to-end availability measurements, automation will be required to simplify the process.



Data Sources Example

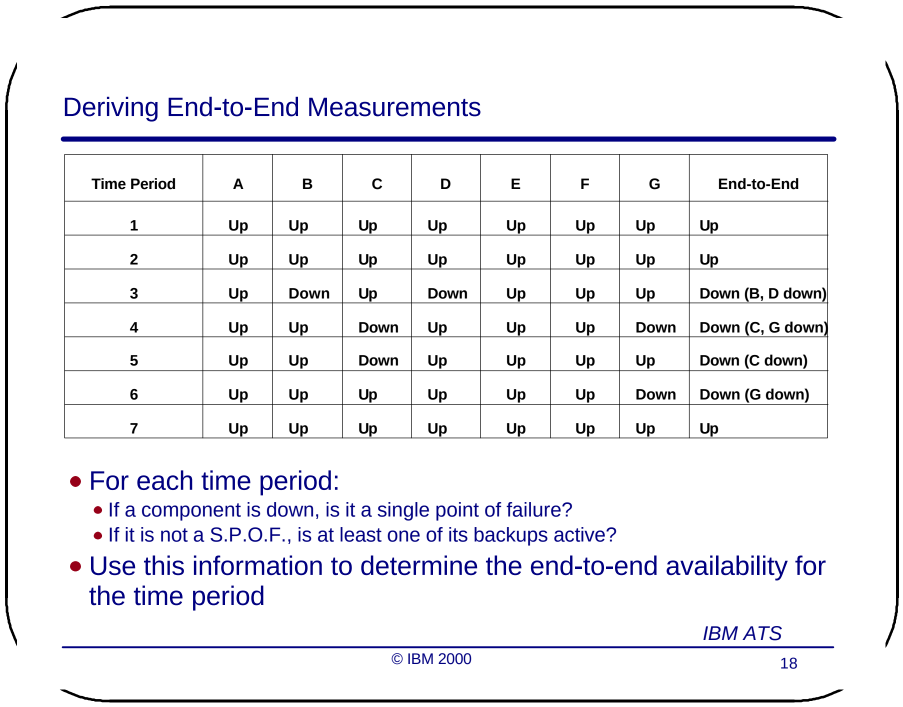
Component	Measurement Data Source	Data Capture Method
SiteA, SiteB Clients	User simulation program running at each site	Observation results stored on workstation and uploaded daily
Router1, Router2, Router 3	PINGs from Network Management Software	Automation sends out periodically and analyzes and stores results
Web Server1 + Windows NT	Heartbeat program Workstation Management Software monitor	Heartbeat result file uploaded daily Management software log scanned daily
Web Server2 + OS/2	Heartbeat program Workstation Management Software monitor	Heartbeat result file uploaded daily Management software log scanned daily
Web Server1 HTTPD Application	Management Software	Application monitors for active process and send exceptions to host as alert
Web Server2 HTTPD Application	Management Software	Application monitors for active process and send exceptions to host as alert
Web Server1 Sockets-to-Host Application	Application	Records status of host connection attempts and results into log, which is examined daily
Web Server2 Sockets-to-Host Application	Application	Records status of host connection attempts and results into log, which is examined daily
LAN-to-Host gateway	Host automation product	Detects and records status changes
VTAM address space	Host automation product	Detects and records status changes
TCP/IP address space	Host automation product	Detects and records status changes
CICS address space	Host automation product	Detects and records status changes
CICS sockets application	Remote command monitor	Sends data query periodically and records results
RAMAC subsystem volumes	Host automation product	Monitors online and offline status
Data	Host automation product	Queries CICS periodically for file status

IBM ATS

© IBM 2000 17

Using our application example, this table shows, for each measured component, the data source and the method that will be used to capture data. Some observations:

- Both products and custom code (heartbeat programs) are being used for measurement and data capture.
- Multiple products and techniques across multiple platforms are being used.
- The captured data initially resides on several platforms, but will be consolidated in one place to produce the measurements. The consolidation is being done once a day.
- Once a data source and capture method is applied to a component, it can be reapplied to all other components of the same type (for example, all address spaces, all web servers). This leverages the first time work effort of finding the data and developing a capture method for a component.



Deriving End-to-End Measurements

Time Period	A	B	C	D	E	F	G	End-to-End
1	Up	Up	Up	Up	Up	Up	Up	Up
2	Up	Up	Up	Up	Up	Up	Up	Up
3	Up	Down	Up	Down	Up	Up	Up	Down (B, D down)
4	Up	Up	Down	Up	Up	Up	Down	Down (C, G down)
5	Up	Up	Down	Up	Up	Up	Up	Down (C down)
6	Up	Up	Up	Up	Up	Up	Down	Down (G down)
7	Up	Up	Up	Up	Up	Up	Up	Up

- For each time period:
  - If a component is down, is it a single point of failure?
  - If it is not a S.P.O.F., is at least one of its backups active?
- Use this information to determine the end-to-end availability for the time period

IBM ATS

© IBM 2000 18

After the data has been collected, the next step is to **derive end-to-end availability from the component data.**

Obtaining the end-to end availability requires applying calculations to the collected data. Using the component relationships and the data collected, the general calculations are done as follows:

- For each component outage, the outage start and end times are used to derive the individual component outage length.
- If a component outage occurs, and the component has dependent components, outages are recorded for them even if there is not direct outage measurement data for them. This can occur when a component goes down but no events are issued indicating that a dependent component is down.
- An end-to-end outage occurs when one or components suffers an outage, and ends when all required components are active. For each time period, the data is scanned to determine if any components encountered an outage. If so, the next step is to determine if:
  - The component is a single point of failure
  - The component is not a single point of failure, but all of its backup components are also down.
- If the two conditions noted in the previous step are true, then an end-to-end outage is recorded. When they are no longer true, the end-to-end outage ends.

- if multiple component outages occur at the same time, the overlapping time is counted for each component outage, but NOT for the end-to-end outage time.

The table shows a matrix where each column represents a component, and each row represents a time period. If any one component is down, an end-to-end outage has occurred. The collected data will show for each time period the status of the components; every time period where at least one component suffered an outage is counted as an end-to-end outage. For time periods 3,4,5,6, an end-to-end outage occurred. If each time period were a minute, the end-to-end outage time would be four minutes. The individual component outage times would range from zero to 2 minutes. This shows the importance of looking at multiple components - an individual component outage time would not have accurately reflected the end-to-end outage time.

In another example, assume three components A, B, and C are required to support an application, and the following data is collected:

Component	Down	Up	Length
A	10:00	10:30	00:30
B	10:15	10:45	00:30
C	14:00	14:30	00:20

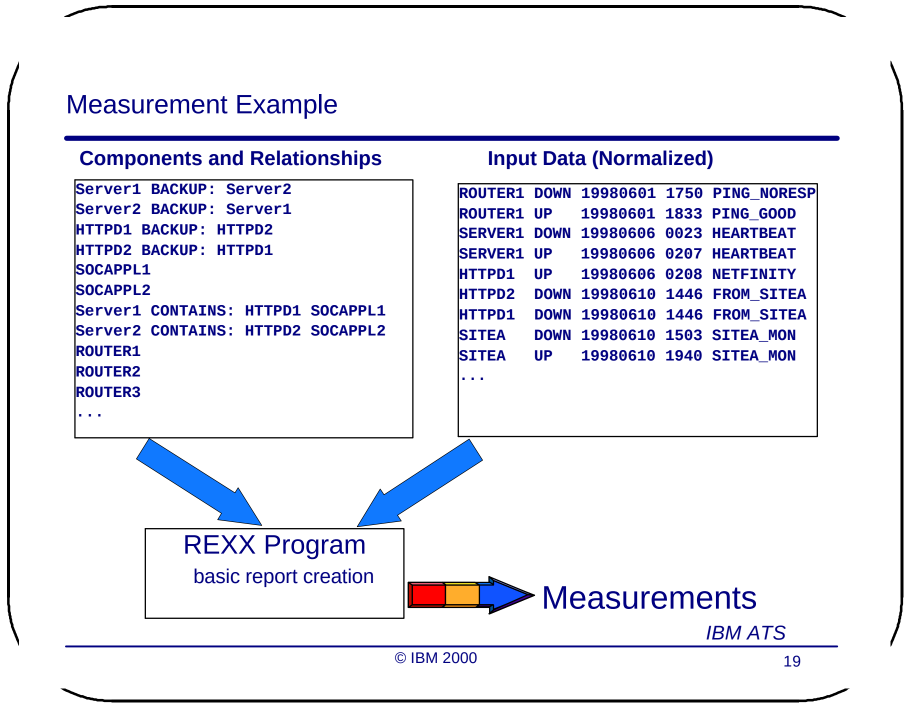
The component outage times for A and B are 30 minutes each, for C 20 minutes. The end-to-end outage is 65 minutes (total time at least one component was down). The availability percentage will depend on the time range that is desired (for a single day, the end-to-end availability would be 95.5%).

The component relationships established earlier can be used to address the following situations that can, if not accounted for, render the measurements inaccurate:

- Handling components with backups, since loss of a component with a backup does not mean that end-to-end availability is lost. The data analysis would check, when a component goes down, if at least one backup is still available. The outage would count against the component, but not against end-to-end availability unless ALL backup components were down. This could still be tracked as a 'degraded' condition, if desired.
- Handling outage 'cascades' of multiple related components. An example: a network communications device goes down, which causes the attached communications paths to fail. Notifications that the paths are lost may or may not be received. If the "dependent" relationships have been identified, the analysis would handle this by recording all measured components that are dependent on the communications device as "down" when the communications device is down. When the communications device comes back up, the analysis marks the attached paths as up, or looks for component records that indicate that the paths are back up. This will not change the end-to-end outage time, since multiple simultaneous component outages overlap and do not affect the end-to-end time.

- Handling user recovery time. Depending on where the end-to-end outage has occurred, the time for the user (or workload) to get back to the state it was at the time of the outage may vary. There is no easy way to determine this time (the user simulation monitoring technique comes closest to doing this); an acceptable action would be to add in some "factor" for a particular component outage to account for the user recovery time. For example, if an application subsystem "up" event is received, an additional five minutes may be required before user transactions can actually be processed.

PAGE 19



Measurement Example

Components and Relationships

```
Server1 BACKUP: Server2
Server2 BACKUP: Server1
HTTPD1 BACKUP: HTTPD2
HTTPD2 BACKUP: HTTPD1
SOCAPPL1
SOCAPPL2
Server1 CONTAINS: HTTPD1 SOCAPPL1
Server2 CONTAINS: HTTPD2 SOCAPPL2
ROUTER1
ROUTER2
ROUTER3
...
```

Input Data (Normalized)

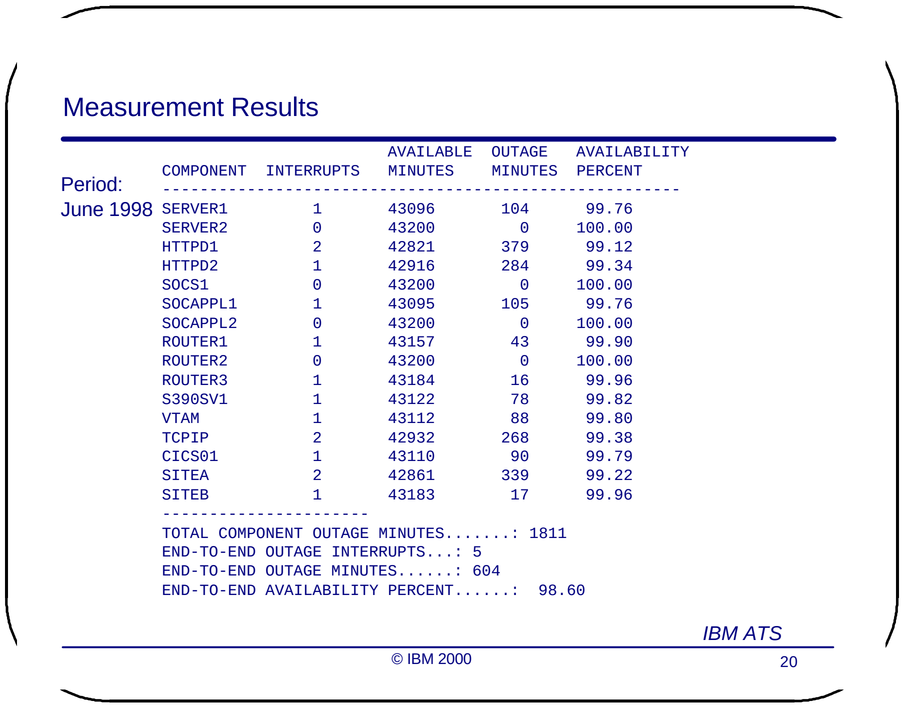
```
ROUTER1 DOWN 19980601 1750 PING_NORESP
ROUTER1 UP   19980601 1833 PING_GOOD
SERVER1 DOWN 19980606 0023 HEARTBEAT
SERVER1 UP   19980606 0207 HEARTBEAT
HTTPD1  UP   19980606 0208 NETFINITY
HTTPD2  DOWN 19980610 1446 FROM_SITEA
HTTPD1  DOWN 19980610 1446 FROM_SITEA
SITEA   DOWN 19980610 1503 SITEA_MON
SITEA   UP   19980610 1940 SITEA_MON
...
```

IBM ATS

© IBM 2000 19

For the example application, this picture gives a snapshot of the measurement process described on the previous page. A REXX program (approximately 400 lines of code) receives two input files. The first input file, on the upper left, describes the component relationships. The second input file, on the upper right, is the component outage data. It has been captured, normalized, and formatted into a common layout, containing the required data highlighted earlier. This allows consistent processing by the program, regardless of the source or capture method. The output from the REXX program is a set of measurements.

PAGE 20



Measurement Results

Period:
June 1998

```
                               AVAILABLE  OUTAGE   AVAILABILITY
COMPONENT  INTERRUPTS   MINUTES    MINUTES  PERCENT
-----------------------------------------------------
SERVER1          1       43096       104      99.76
SERVER2          0       43200         0     100.00
HTTPD1           2       42821       379      99.12
HTTPD2           1       42916       284      99.34
SOCS1            0       43200         0     100.00
SOCAPPL1         1       43095       105      99.76
SOCAPPL2         0       43200         0     100.00
ROUTER1          1       43157        43      99.90
ROUTER2          0       43200         0     100.00
ROUTER3          1       43184        16      99.96
S390SV1          1       43122        78      99.82
VTAM             1       43112        88      99.80
TCPIP            2       42932       268      99.38
CICS01           1       43110        90      99.79
SITEA            2       42861       339      99.22
SITEB            1       43183        17      99.96
---------------------
TOTAL COMPONENT OUTAGE MINUTES.......: 1811
END-TO-END OUTAGE INTERRUPTS...: 5
END-TO-END OUTAGE MINUTES......: 604
END-TO-END AVAILABILITY PERCENT......:  98.60
```

IBM ATS

© IBM 2000 20

A useful set of availability reports contain at least the following information:

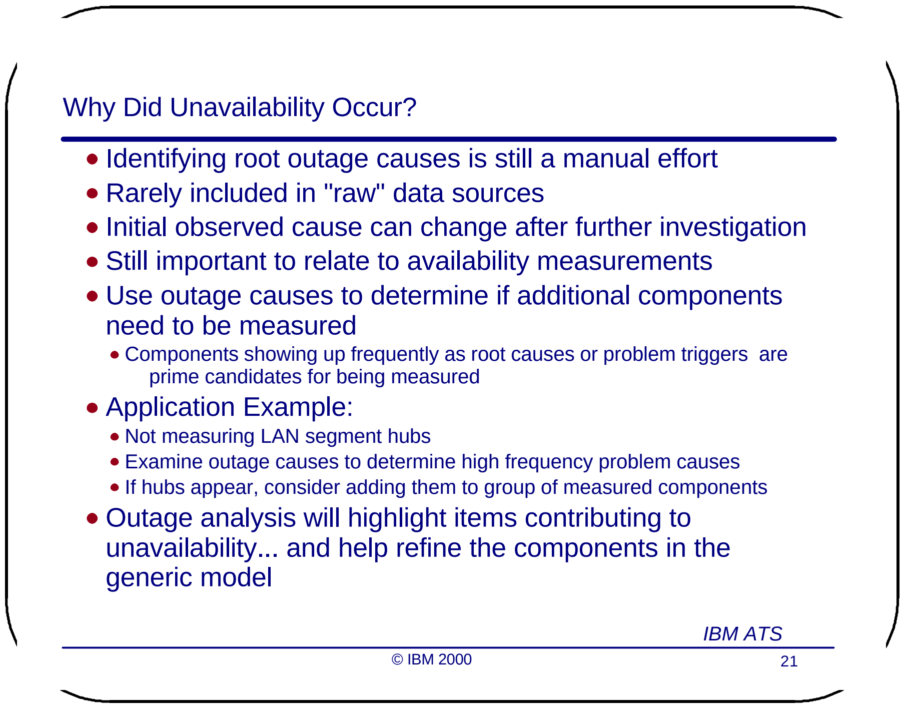
- End-to-end availability for measured applications
- Component availability for measured components that support the applications
- Frequency and duration of outages for the end-to-end application and the individual components
- Frequency and duration of outages by outage categories (this requires relating availability and problem/change data together)
- Impact of an outage in terms of affected users and/or workload

This report shows a subset of that information. It is the output of the REXX program used in the previous presentation page. The reports shows every measured component for the example application, the number of outages (interrupts), The available and outage minutes, and the availability percentage. The last three lines show the information for the end-to-end application. Note that simply totally the component outage minutes does not accurately reflect the end-to-end minutes; this is due to the component relationships and any overlapping component outages. Also note that any individual component measurement does not accurately reflect the end-to-end measurement. This report can reveal which component had the highest number of outage minutes. This might also be the component with the biggest impact to availability - if it is a single point of failure. Otherwise, a more detailed look at the data will be required.

The advantages of reporting availability in more than "percentage available" will be covered in the next step. The information in this type of report gives both a useful snapshot of how things stand today, and long-term information for spotting trends and taking appropriate actions.

There are many tools available to report this information; installations have used anything from PC spreadsheet programs to full blown reporting products and data repositories. Whatever route is chosen, the reports must be used in conjunction with other systems management data from problem, change, and performance management processes to identify required improvements.

PAGE 21



Why Did Unavailability Occur?

- Identifying root outage causes is still a manual effort
- Rarely included in "raw" data sources
- Initial observed cause can change after further investigation
- Still important to relate to availability measurements
- Use outage causes to determine if additional components need to be measured
  - Components showing up frequently as root causes or problem triggers are prime candidates for being measured
- Application Example:
  - Not measuring LAN segment hubs
  - Examine outage causes to determine high frequency problem causes
  - If hubs appear, consider adding them to group of measured components
- Outage analysis will highlight items contributing to unavailability... and help refine the components in the generic model

IBM ATS

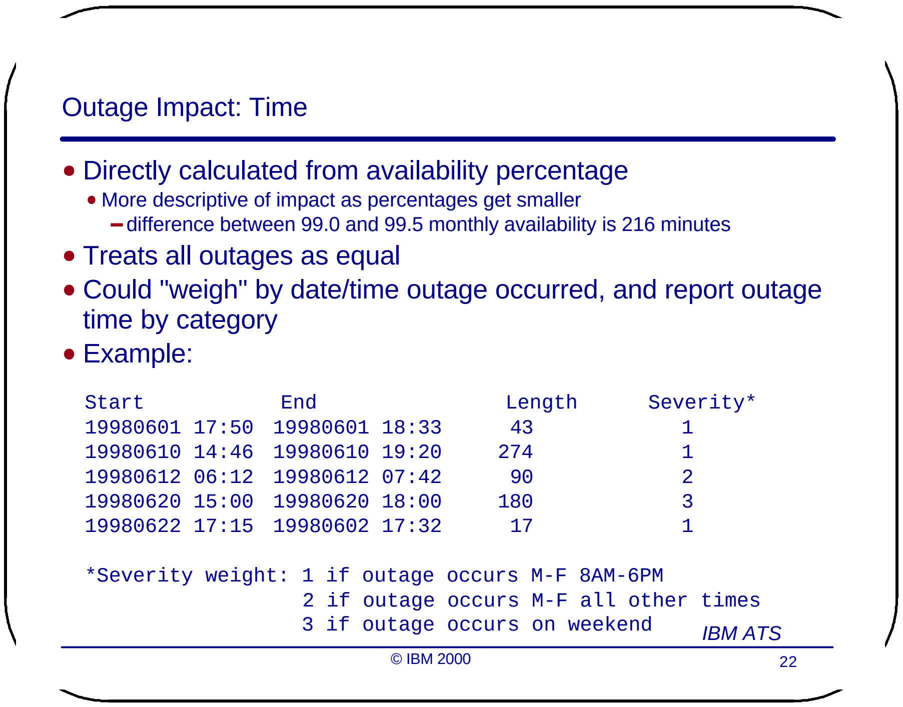
© IBM 2000 21

The final step, **Identify unavailability location(s) and impact** , must be done to bring about improvement. Once unavailability has been identified, the cause of that unavailability, or outage, must also be identified to determine how that type of outage can be avoided or eliminated in the future.

Finding the real cause is an activity commonly called "root cause analysis". It tries to identify, for an outage, the triggering event that initiated the outage and the reasons the triggering event occurred. It can be days, weeks, even months after an outage occurs to determine the root cause of that outage. While some sources of availability data may identify the triggering event, determining the actual root cause is still a manual process, carried out as part of the problem management process.

Once the outage cause is determined, it must be recorded (usually manually for reasons discussed above). In the application example, the TCPIP host address space component had a high impact on unavailability, but the measurements did not show what specific problems caused the outage minutes. If the problem tracking/resolution process includes the use of availability data, when the proper cause of the outage is determined it can be recorded and associated with the measurement information. This allows outage cause reports to be consistent with availability reports.

Another benefit of finding and recording outage causes is to identify additional components that should be measured. In the web application example, the user LAN segments are physically an ethernet hubs. They are not being measured. Recording outage causes and looking at the frequent causes or triggering events will show if the hubs are a problem. Using this information can lead to the hubs being identified as critical components and to include them in the measurement steps. By using the measurement data long with outage analysis, components contributing to end-to-end unavailability will be highlighted, and will help better identify what should be measured.



Outage Impact: Time

- Directly calculated from availability percentage
  - More descriptive of impact as percentages get smaller
    - difference between 99.0 and 99.5 monthly availability is 216 minutes
- Treats all outages as equal
- Could "weigh" by date/time outage occurred, and report outage time by category
- Example:

```
Start           End                 Length      Severity*
19980601 17:50  19980601 18:33        43            1
19980610 14:46  19980610 19:20       274            1
19980612 06:12  19980612 07:42        90            2
19980620 15:00  19980620 18:00       180            3
19980622 17:15  19980602 17:32        17            1

*Severity weight: 1 if outage occurs M-F 8AM-6PM
                  2 if outage occurs M-F all other times
                  3 if outage occurs on weekend
```

IBM ATS

© IBM 2000 22

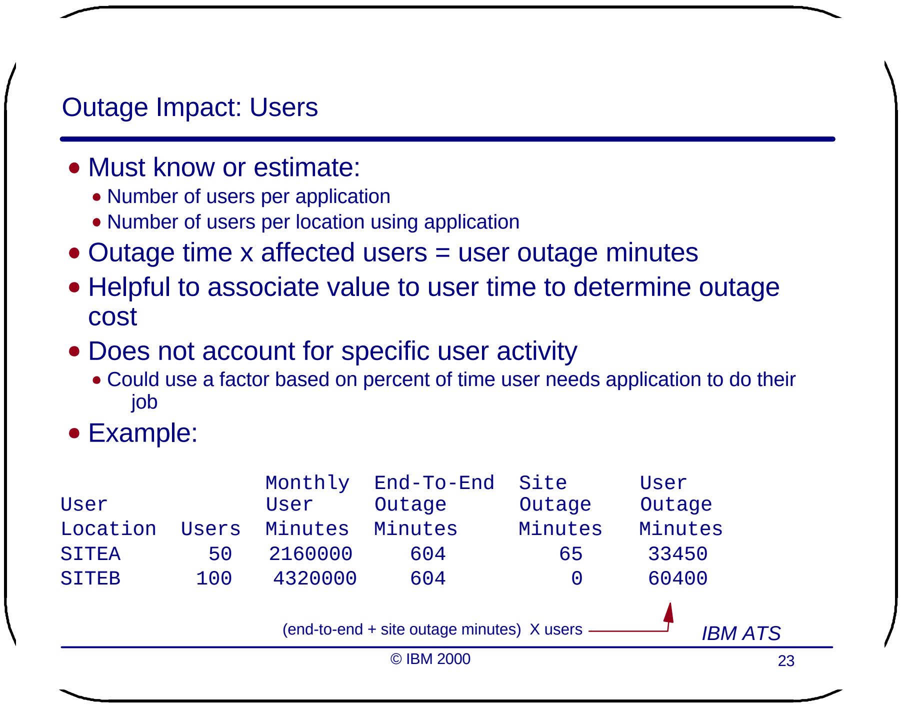
Quantifying the impact of an outage needs a better metric than just percentage. Saying "it is 60 degrees outside" is meaningless without a context - where is it 60 degrees? In Edmonton in January? In Aruba in July? on Mars? Each of these will have a different impact and require different (if any) actions. Likewise, simply stating "availability is XX percent" is meaningless without a context to reflect what the impact of unavailability is and if that impact is of concern. One metric that should always be used is **outage frequency**. Three other metrics to consider are covered on this and the following two presentation pages.

1. **Time.** This is tabulated and used to calculate the from the availability percentage**.** The formula to get back from the availability percentage to time is:

```
Outage time = Total period time * (1-availability percentage)
```

In the web enabled application example, the end-to-end outage time for the week was calculated to be 604 minutes.

The problem with using time without any qualifier is that it treats all outages as equal. A ten minute outage at 2 PM is considered the same as a ten minute outage at 2 AM. One way to address this is to assign weights to different time periods to indicate the importance to getting business work done. Then the outage time by "weight" would be reported to better reflect the unavailability impact. For the application example, the page shows the start time, end time, and length of the five end-to-end outages. A severity weight has been assigned based on when the outage occurred. Using the severity weight will identify with outages were more significant and help prioritize where root cause analysis and improvements are applied.



Outage Impact: Users

- Must know or estimate:
  - Number of users per application
  - Number of users per location using application
- Outage time x affected users = user outage minutes
- Helpful to associate value to user time to determine outage cost
- Does not account for specific user activity
  - Could use a factor based on percent of time user needs application to do their job
- Example:

```
                    Monthly  End-To-End  Site      User
User                User     Outage      Outage    Outage
Location   Users    Minutes  Minutes     Minutes   Minutes
SITEA         50    2160000     604         65      33450
SITEB        100     4320000    604          0      60400
```

(end-to-end + site outage minutes) X users

IBM ATS

© IBM 2000 23

2. **Users.** One of the reasons, when defining component relationships, to identify which users (or groups of users) use an application or connect over a path is to determine outage impact based on the number of users affected. A more detailed view beyond this is the number of user outage minutes, which can be calculated using the outage minutes:

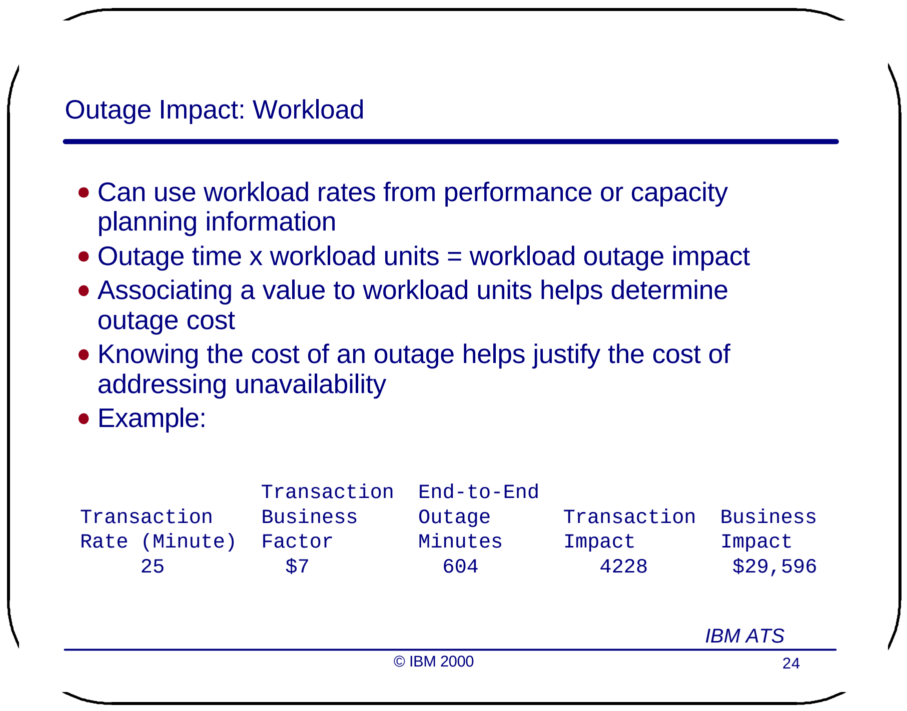
```
user outage minutes = (number of affected users) * outage minutes
```

For the application example, the page shows the user outage minutes by site. Note that, in addition to end-to-end outage minutes, Site A also incurred outage minutes that did not affect Site B; the component relationships revealed this information.

Using the users or user outage minutes metric can provide a business cost of an outage, in terms of lost user time, productivity, or satisfaction. If a value can be associated with the users time, then the cost of an outage in user terms can be measured. This cost can then be used in evaluating any investments to address outages.

The drawbacks expressed by some organizations to use user measurements tend to be concerns over who is actually using an application ("we don't know how many people are active at a given time") and what their time is worth ("we can't calculate what these users value it, and even if we could, how do we know what they are doing? they could be on break when an outage occurs"). It is still recommended that some attempt to estimate this be done. Unless a starting point is picked, the issue will never be addressed. This information may take some research to gather; initially may even be a gut feel - but is needed to start viewing outage impacts in a business context.

PAGE 24



Outage Impact: Workload

- Can use workload rates from performance or capacity planning information
- Outage time x workload units = workload outage impact
- Associating a value to workload units helps determine outage cost
- Knowing the cost of an outage helps justify the cost of addressing unavailability
- Example:

Transaction Rate (Minute)	Transaction Business Factor	End-to-End Outage Minutes	Transaction Impact	Business Impact
25	$7	604	4228	$29,596

IBM ATS

© IBM 2000 24

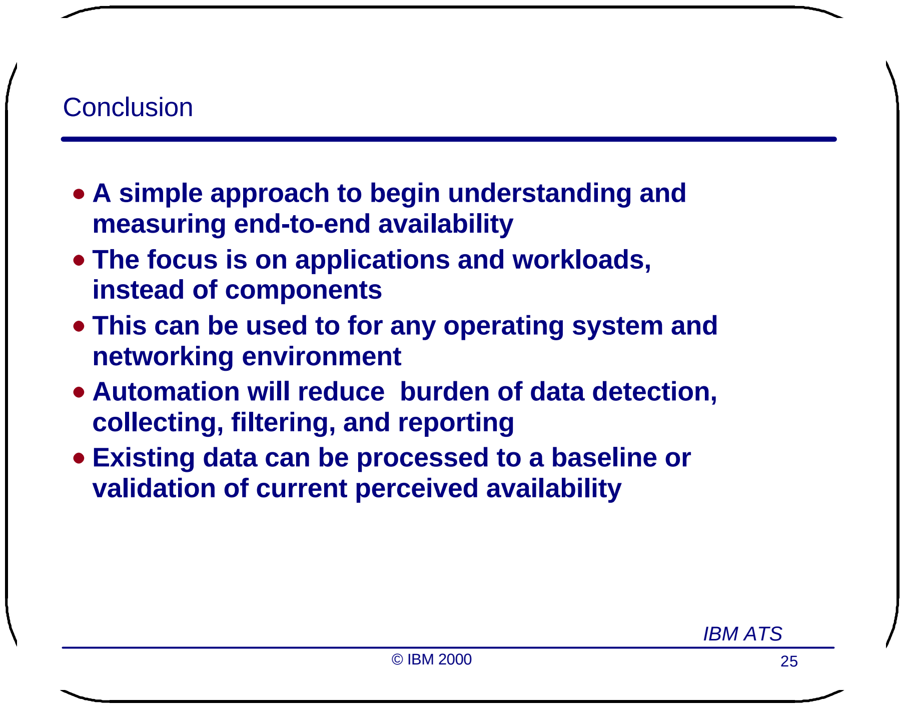
3. **Workload.** If workloads (and workload rates) have been identified for an application, the impact of an end-to-end outage in workload terms can be calculated using the outage minutes:

```
workload outage impact = units of work per minute * outage minutes
```

For the application example, the page shows a transaction rate of 25 per minute, with a transaction business factor of $7. With 604 end-to-end outage minutes, the result is 4228 impacted transactions at a business cost of $29,596. As with the user measurement, this provides information on the business cost of an outage if a value is associated per unit of work. The business impact is workload lost or workload delayed (meaning it has to be rerun at a later time).
Drawbacks similar to the "user cost" drawbacks have been expressed for this measurement. However, if an installation is doing proper performance monitoring and capacity planning, they already have the information on work unit transaction rates, and this information can easily be combined with the measurement information. Likewise, the nature of the application reveals the cost of lost/delayed work. For example, an automated teller application lost work may me lost fees or penalties; a customer ordering application lost work may mean lost revenue, or higher cost per transaction. It is important to extend the context beyond just time (such as to the user or workload measures described above).

Knowing the business cost of an outage is the first step towards justifying improvements in availability. It is not worth spending X currency units to justify a .5X outage cost (unless there are other less tangible benefits such as customer satisfaction or political survival). On the other hand, it is very much worth spending X currency units if the benefit is 10X. Using this outage impact data in the proper way will help identify the worthwhile availability improvements.

PAGE 25



Conclusion

- A simple approach to begin understanding and measuring end-to-end availability
- The focus is on applications and workloads, instead of components
- This can be used to for any operating system and networking environment
- Automation will reduce burden of data detection, collecting, filtering, and reporting
- Existing data can be processed to a baseline or validation of current perceived availability

IBM ATS

© IBM 2000 25

These steps are not a total methodology; they are a simple approach that can be undertaken to begin measuring end-to-end availability. One can start simple, and expand its scope without having to redo much of the previous work. The benefits of using these steps include the following:

- It moves the availability metric focus from components to end-to-end applications, to get a more accurate picture of how availability affects the users and business functions supported by applications. A side benefit is that it gets I/T support and users looking at the same information in a more objective manner.
- It required a minimum amount of data and components to get started, and can be easily expanded as an environment gains experience using these steps.
- It can be applied independent of any particular operating system or network environment. An installation whose largest application platform is a single workstation-based server can use it, as well as an installation that has dozens of mainframe based servers. The end-to-end availability of an application running on different host/distributed configurations can be compared, to show which environment is better suited to the availability requirements of an application.
- It takes advantage of automation and products that are already present in the installation remove the manual burden of data detection, filtering, collection, and reporting.
- It relates data on availability, impact, and outage cause information in a variety of ways, and can, by applying various metrics, provide a great detail of availability and outage information that is useful for inclusion with other systems management information.

- It can analyze the current environment and determine a baseline of availability. Knowing this baseline can be helpful in evaluating the impact of infrastructure and application changes to see if actions can be taken to improve availability, if availability did/did not improve after changes, or what infrastructure components have to be addressed to provide higher end-to-end availability for current or new applications.

Measuring availability is one - and not the only - activity that must be done to manage the I/T environment over the long run. Understanding what is being achieved and what is possible end-to-end, and using these steps to accomplish this task, will be beneficial for any organization and any application environment.

FOR FURTHER INFORMATION ON AVAILABILITY MANAGEMENT...

There are several publications that will assist in understanding and carrying all of the steps required for availability management:

- Continuous Availability System Design Guide (SG24-2085)
- Systems Analysis For High Availability (GG22-9391)
- So You Want to Estimate the Value of Availability (GG22-9318)
- Systems Outage Analysis (GC20-1871)

IBM Education and Training offers courses on Availability Management to help in the understanding of the overall process and the planning considerations that are necessary. In addition, the IBM Consulting Practice conducts engagements for customers to assess the Availability Management process that is being used or considered, and to identify what steps should be implemented and/or where improvements can be made.