Almost all major IS Departments are faced with monumental efforts in preparing for required Year 2000 system changes. We will soon be counting the time to Year 2000 in months and days, not years. This paper will detail the critical tasks that must be performed to insure a safe transition.

The authors will discuss the prioritization of applications, types and intensity of testing, and business process risk assessment. In addition, we will discuss contingency planning to handle post 2000 errors when (not if) they occur. Finally, the issue of the capacity needed to support the Y2K testing efforts and production requirements for these final months and beyond 2000 will be covered.

1.0 Introduction

Time is running out. Difficulties caused by the Year 2000 challenge are already occurring and many systems dealing with future dates in the areas of finance, insurance, banking and credit cards have encountered problems. Planning efforts have begun and an identification of potential risks is underway. The real challenge for most corporations is that there is not enough time left to fix and test all the changes necessary to be fully Year 2000 compliant.

The scope of the Year 2000 challenge is pervasive across the entire Information Technology industry. A data mismatch can exist in any level of hardware or software, from microcode to applications, in files and databases. It is not limited to mainframe systems and can be present on any platform.

Unlike past data processing activities, there is no way to bring this project in late; there is a solid deadline, December 31, 1999. In fact, by one estimate, October of 1997 is the last point at which Year 2000 repairs could start with a reasonable probability of finishing before 2000. Obviously, those companies which did not get started by year end 1997, will need to limit their focus to just key mission critical applications to minimally insure that those systems can be successfully converted. All IT professionals need to take a step back and reassess their Year 2000 plans to determine whether all systems can be converted within the remaining timeframe. The major components of this new strategy will include development, testing and conversion efforts as well as the production implementation of the modified Y2K compliant software. Contingency planning will be required for those applications that cannot be addressed within the window.

We will discuss the capacity requirements to support the conversion effort, from testing through production implementation and provide methodology for sizing the test and production
requirements to support this effort. We will also review the management strategy for contingency planning and risk assessment.

2.0 Background

Since the beginning of data processing, the year has been stored as a 2 digit field. At the end of the 20th century, many software applications that use dates for time-sensitive calculations will stop working or produce erroneous results when the year switches to 00. All industries are affected by this problem.

The solution is not trivial and will require significant resources to solve it. There is no longer enough time nor sufficient resources to make all systems and applications Year 2000 compliant. Conversion efforts will require significant amounts of code to be modified, and its development, testing and production implementation will have a resulting impact on system performance and capacity. Capers Jones, a leading expert discussing the economic impact of the Year 2000 problem, indicates that failure to repair the Y2K problem in a cautious fashion, allowing for sufficient testing and performance optimization, will degrade overall data center throughput by more than 20% [Ref.5].

Compounding this staffing issue is the need for additional computing resources to support the increased testing that will be required. Hardware will need to be upgraded to support the extra testing, development and production load. We anticipate at least a 20% increase in hardware resources necessary to support the effort.

3.0 Planning and Risk Assessment

The year 2000 challenge could not be solved in a few days or a few months and most experts agree that work needed to commence by October 1997 to resolve the problems the organization will encounter [Ref. 1,5,6,8,9]. By this point in time, organizations should have completed a review of their application portfolios and have detailed knowledge of the systems, software and hardware that they use. They should have an understanding of system interdependencies, service levels and impact to the user community. This information is crucial to insure that the organization can have a functioning, operationally Year 2000 ready computing environment before any disruption can be caused by 2-digit-year data.

The Year 2000 solution requires a four step process:

1. Impact Analysis. Perform an inventory of applications and databases in terms of whether there is any potential exposure for Year 2000 problems.

   Note: This step should have already been completed and should serve as a foundation for reassessing the plan and configuring the test and production environment.

2. Planning and Scheduling. Project level planning for applications to make them year 2000 compliant.
3. **Conversion.** Implementing the strategy and restructuring the source code and databases and files.

4. **Testing and Implementation.** Testing and implementation will account for 40% of an organization’s total project [Ref. 3].

But now two more steps must be added to the process. A **triage phase** must be added to reassess the entire application inventory and test plans. The focus must be to complete mission critical applications and their application and system dependencies. A **contingency phase** must be introduced to address those systems which cannot be successfully converted in the remaining time window.

5. **Triage Phase.** Reassess all plans and schedules and assess business risks associated with each system.

6. **Contingency Planning.** Detailing the requisite plans for system remediation management for those systems which cannot be converted in time.

### 3.1 Impact Analysis

The impact analysis provided organizations with an identification of the frequency with which programs were executed, files used and the resulting exposure to Year 2000 problems. The exposure identified could be: none, cosmetic such as in report presentation, or date sensitive such as in logic routines requiring calculations or date validation. The later systems require the most effort since their logic needs careful examination.

The risk assessment phase identified those factors critical to the success of your business and their relative priority. It is this assessment which must now be reexamined during the triage phase.

The impact analysis was an important first step in determining the scope and magnitude of the effort. A significant amount of code rework is required to complete the Year 2000 transition. It is not merely a problem that can be fixed by expanding the data fields. Changes must be made to dictionaries, data bases, files, programs, etc. There are automated tools for performing this analysis and estimating the time and resources required for the conversion effort. It should be noted that the tools and solutions are dependent upon the languages and databases utilized.

### 3.2 Planning

Management's understanding, support, and commitment is critical to addressing the challenges of Year 2000 issues and to obtain the necessary resources to meet functional requirements. A separate task force headed by a knowledgeable manager to address budgetary, personnel and hardware resource requirements, cross-department and cross-divisional requirements, as well as scheduling and project management issues is essential. The project manager is responsible for critical project aspects such as overall scheduling, coordinating, and setting a consistent methodology through all project phases.

The planning effort includes prioritizing the project steps and determining the strategy for restructuring, redevelopment or replacement of the application. Programming possibilities include developing a bridge between the original files with 2 digit dates and the programs to make the system Y2K capable. Another approach is to use automated tools, called date-server software which change only source code while leaving the two-digit year fields in place. Gartner Group believes that organizations will wind up using a combination of date conversion routines and date servers.

Planning for the Year 2000 will impact all aspects of your IS environment and requires an identification of the deliverables and associated schedules for hardware, software, documentation,
training, maintenance, operations, and administration. For each task, persons or organizations responsible such as, customers, management or vendors must be identified. The task force must include representation from End Users, Systems Analysts, Performance and Capacity Analysts, Applications, Operations and Quality Assurance Groups to be effective. The group must determine how to interface with and obtain information from outside vendors and consultants and integration services providers.

The effort involved in terms of costs, completion times and exposures and risks must be quantified for each task. Technical considerations must be taken into account and resource assessments made based on criticality of task and priority. Risk factors must be evaluated in terms of:

- Business priority
- Technical priority
- Complexity of the task
- Resource/time constraints
- Length of project
- Critical development skills

Those tasks and applications that are critical and impact your business, such as legal compliance issues, and those having the potential for financial penalty if unresolved, will need to be tackled first. In the triage phase, you will need to ensure that these systems and applications either have been addressed or can be completed in time.

For all applications, the scope and responsibility of migrating the affected data must first be determined including whether there are any external exposures such as those beyond the control of the enterprise. Once this is accomplished, the Year 2000 plan can be defined to include overall methodology, procedures for changing data elements, programming standards and conventions, application development system environment, hardware and software, and development and test procedures.

**3.3 Conversion**

Once the Year 2000 plan is put in place, converting software programs for Year 2000 readiness can begin. The first step requires a thorough understanding of your computing environment and an inventory of all software programs within that environment. This inventory will help to:

- Analyze the programs for definition and movement of date-related data elements and the use of date-related calculations and manipulation.
- Identify Year 2000 exposures.
- Track and control changes to software and more readily monitor its performance.
- Test the new Year 2000 versions of the software.

The conversion effort should be thought of as similar to or greater than any previous application development effort. The effort includes design and specification, coding, unit testing, integration testing, regression testing, full system testing, and finally, user acceptance testing. With each of the options, restructuring, redevelopment or replacement will each have substantial testing requirements to ensure that the systems function correctly. There are automated test tools that are available for doing this. A list of tools and companies performing consulting activity can be found in Kappelman and Scott, “What Management Needs to Know about the Year 2000 computer Date problem”, Com.Links Magazine [Ref. 8] as well as ACRs Resource Guide.

Locating date-related data and code is itself a significant part of the total work effort. The most complete method starts with an inventory of every programming entity used in Information Systems. Once compiled, each program should be individually reviewed. There are Developer Tool
Kits available to provide more systematic automated approaches to locating 2-digit-year data and date-related code.

Other options include the creation of user and system surveys to obtain an assessment of source code compliance. For systems developed in the past 5-10 years, documentation may exist to aid in the identification of date related code including:

- Statements of work
- Existing studies about the current system
- Software development standards and process documents
- System requirements specifications
- System design specifications
- Program specifications
- User instructions and procedures
- Data dictionaries

An inventory of all code which must be modified for Year 2000 should be stored in a centralized repository for review. All code must be inventoried and tracked and its relationship to other code determined. A total count of the lines of code will assist in determining how many and what type of resources will be necessary for making the changes. Organizing the code by application can assist in the development of the detailed plan. The code inventory will also contain an indication of which applications exchange data and should indicate the production job names and transactions to which they are mapped.

The survey process seeks to collect system data in a standard format and can augment much of the information obtained from some of the automated tools. Each application must be inventoried to include system identifiers, descriptions, platform and language components, interfaces, owners/users/maintainers, production job names and other relevant information critical to any prioritization and scheduling effort. This inventory of information is essential to planning and scheduling the deployment of development, test and production. But it is also a critical input to sizing the environments necessary to support the Y2K conversion effort from development through regression testing and production implementation.

3.4 Testing

In order to be Year 2000 compliant applications and systems must be able to perform the following:

- Correctly process dates before and after January 1, 2000.
- Recognize Year 2000 as a leap year
  - Recognizing 2/29/2000 as valid
  - Recognizing short Julian date 00060 to represent 2/29/2000
  - Recognizing short Julian date 00366 to represent 12/31/2000
  - Calculating the number of days between 3/1/2000 and 2/28/2000 as 2.
- Accept and display dates unambiguously
- Correctly process logic dates that are used for non-date functions such as
  - Random number generators
  - Information archiving functions
  - Naming conventions
  - Passwords with expiration dates
- Critical midnight crossings
  - The basis millennium transition-December 31,1999 to January 1, 2000
  - The 1st leap year midnight transition-February 28, 2000 to February 29, 2000
  - The 2nd leap year transition - February 29, 2000 to March 1, 2000

Testing preparation will require the creation of an environment to support the applications and systems which will be performing:

- creation of scripts and data to verify processing of all date functions across different time points.
- data conversion for some applications
- unit testing of code changes
• systems testing of code changes
• stress testing of mission critical applications

In order to properly test the changes being proposed for Y2K, the most reasonable approach is to install an isolated, non-production system with a duplicated image of your system and application software. In large systems, this could be an LPAR (logical partition) of the mainframe, or for other platforms it could be a dedicated separate machine, segregated from other systems. This segregation is necessary to isolate system clock changes from existing work and to be able to test the modified code. Changed system date and times to the future will be necessary to test code compatibility beyond January-01-2000.

Most organizations will not be prepared to undergo the expense and effort involved in replicating hardware resources to support a segregated Year 2000 testing environment. But at this point, with the millennium a little over one year away, it is the safest approach. Failure to establish a controlled testing environment exposes the organization to serious capacity, performance and reliability issues. It is recommended that every effort be made to at least logically isolate the test environment and limit its resource impact on production.

3.5 Triage Phase
The purpose of this phase is to determine where to focus your limited resources. During the initial inventory assessment phase you were able to determine the extent of your Year 2000 exposure. You should therefore already have an estimate of both the number and size of the systems which are impacted. It is critical to refocus your efforts on what can be accomplished within the specified timeframe and with the available resources. You need to determine what you can do now, what you will do in the future and what you will not do at all. Focus must be placed on those systems with the greatest risk of causing Y2K problems in mission critical applications.

The triage phase requires that you reassess your Y2K strategy and determine what can be completed within the remaining timeframe. The assessment involves the following activities:

• Assess business risks associated with each system
• Assign a business priority to each system or inventory element
• Define scope by determining which systems will receive Year 2000 detection and correction efforts.
• Develop plans, costs and schedules for detailed assessment and obtain management approval

At this point it is critical to reevaluate your organization’s business plans in light of the resource demands imposed by the Year 2000 conversion effort. Essentially you need to address the plans along the lines of triage and identify the mission critical systems, those of minimal importance and the middle tier. If there are insufficient resources to fully address the middle tier, relative priorities must be reestablished to further quantify that portion of work that can be addressed given the available resources. As a result of this analysis, you may in fact find it necessary to increase resources.

A Triage Swat team to reassess the business risks and examine the portfolio inventory should be formed. The group should include senior managers, staff and other key data management personnel who will be responsible for making decisions and must have authority to effect priority changes of applications and plans for resolution and implementation.

3.6 Contingency Planning
Although everyone is working diligently to ensure that a high percentage of Y2K and related problems will be resolved in time, we must
anticipate that some things will be overlooked, ignored or not completed on time.

There will not be enough time and/or resources to fix everything. During the triage phase, low priority systems were identified and determined to be ignored. Some of the medium priority systems may not be thoroughly tested and even mission critical systems that were tested could have errors.

Contingency planning should address the plans for remediation management. It will be prepared by system users in consultation with developers, operators and systems personnel and should include actions to be performed in the event of an outage.

The plan should contain detailed approaches for alternatives to achieve mission goals in the event of outages or application failures. The plans must identify the roles, responsibilities, and authority of personnel. It must allow for training and testing of plans prior to the outage and procedures for invoking and operating in contingency mode.

From a capacity management standpoint, we must review and understand the contingency plans and project the additional capacity required either in-house or off-site to support this eventuality.

4.0 Capacity and Sizing Considerations for Year 2000

The Y2K issue has serious ramifications for Capacity Planning and Performance beginning now and extending beyond the year 2000. The problems manifest themselves as follows:

- Computing resource capacity
- Performance of applications relating both to CPU and DASD
- Scheduling of development, testing and production systems in the data center.
- Sizing the conversion, development, test and production environments.

These issues must be addressed for both mainframe based legacy systems and client server applications. Testing complexity is compounded by the sharing of data across multiple systems. Synchronization and inter-system testing of interdependent mission critical systems will be of paramount concern and will drive the size and complexity of the required test & production environments.

The capacity and performance impact will exist from now until the Year 2000 and continue after the Year 2000. The impact on capacity, testing and production performance is discussed below in terms of the ramifications prior to the Year 2000 and post Year 2000.

The primary activity of the Year 2000 effort prior to January 1, 2000 is obviously the conversion of existing applications to Y2K compliant formats. The potential impacts will be felt in the following areas and are discussed below:

1. Sizing the Conversion & Development platform.
2. Test system capacity requirements
3. Production impact as applications are migrated to Y2k.
4. Application tuning efforts

4.1 Sizing the Conversion & Development platform.

While we all have some experience with development environments, the conversion effort may be a more unique component of the Y2K process. This is not something we engage in during day-to-day operations to the level that we will in this project. With Y2K, the magnitude of the task involved in isolating and identifying code that requires change, affecting that change or instrumenting techniques to avoid the change is unparalleled in our experience. We will have to
rely on a host of vendor products to automate a very tedious and rote process to complete this phase as quickly as possible. We expect that many corporations will assume that this category of work can be performed within the existing corporate test facilities. But since over 50% of the application portfolio will require some change, the analysis and conversion effort will be of sufficient magnitude to warrant a separate test environment.

It is best to rely on vendor guidelines in determining the environment necessary to support the scoping and analysis effort. Given a detailed inventory assessment, one should be able to determine resource requirements for each group of applications using vendor guidelines. Based on a review of project timelines and an evaluation of those applications that need to be converted concurrently, one can project the resource requirement for each application group. Applications requiring concurrent conversion should have been determined based on shared database/files and/or common program modules. The maximum resource estimate should be used to size this environment.

The most significant task to be accomplished is that of data conversion. Corporations embarking on full program and data conversion will require sufficient CPU and storage resources to support this effort. It is possible that every database in-house will require conversion or at least, review. Clearly, this effort is not trivial and this activity must be carefully planned and executed. The planner must be aware of the temporal issues of synchronizing the updating of executable code with converted data, as this will impact the concurrent testing that must take place.

In the total conversion scenario, the supporting databases for each application need to be in place at the same time that the code is implemented. This task has two parts - the data conversion effort and the requisite space for storage of data. An increase in CPU requirements can also be expected for converted code, and will be discussed in the section on production environments.

The data conversion effort must take existing files and create new files with expanded date structures. The planner needs to determine processor resource requirements, execution times and the size of the resulting databases. The most effective approach is to develop and execute controlled benchmarks for key applications and databases. Selected subsets of work are run through the conversion process and pre and post execution results are reviewed. This effort will yield the execution resource and time requirements for a known quantity of data (i.e. number of records) that can be extrapolated to determine requirements for the complete database. Where applicable, the planner may be able to use the same rule of thumb ratios to predict the requirements for several similar database types. In a similar fashion, the size of the output database can be compared to the input, to develop a factor to be applied in estimating total data storage requirements. This sizing effort must encompass both mainframe and client/server implementations and, where appropriate, the necessary network resources connecting these platforms must also be addressed, particularly increased bandwidth requirements. All of these can be predicted from data obtained from controlled benchmarks.

Total conversion implies that all dates that are currently stored with a two-digit year will be expanded to a four-digit year. While the technical issue is not significant, the logistics and resource needs are potentially enormous. The Year 2000 implementation plan will identify the expected conversion timetable for all applications.

There must be sufficient computing resource to convert all of the data that needs concurrent conversion within the specified window. In addition, there must be sufficient disk storage to support both the source and target databases until it is felt safe to free up the original space.
Sizing the Y2K development platform is no different than sizing any other development effort. There will be a number of programmers updating code. This effort can be simplified considerably if an established resource profile for development users exists. The Y2K plan should provide a detailed time sequence of concurrent application development as well as a projection of development manpower requirements by month. Utilizing the historical development resource profile and the Y2K plan, one can project the requisite CPU, IO and network resources needed to support the development effort.

In the absence of historical information, you will need to first characterize the typical development user. This can be accomplished by benchmarking the activity of developers performing various functions such as editing, compiling, linking and module test activities. Performance measurement data collected for these benchmark runs can be used to develop typical resource profiles. Multiplying these resource estimates by the number of staff who will concurrently perform Y2K development tasks will yield the composite requirements for the test environment. Keep in mind that this effort can and probably will take place on multiple platforms across the enterprise. Be sure to size the mainframe and server requirements for this effort as well as the requisite network resources as they apply.

4.2 Test system capacity requirements

It is estimated that at least 50% of the enterprise’s application portfolio will require some form of code and database changes. Therefore, there a significant amount of code, logic and files will need to be modified in order to conform to Y2K date requirements.

The capacity required to support the system will be dependent upon:

- the number of applications that must be tested.
- the timetable for testing the systems.
- inter-system dependencies including the need to interface between mainframe and client server environments.
- the number of applications that can be tested simultaneously.

For mainframe based legacy systems, the test system will require a separate processor environment and isolated DASD. Additional disk will be necessary to support archival, scratch and reload of databases throughout the testing phases.

Networking and open, distributed computing allow data to flow from site to site, system program to application program, etc. Sufficient testing of inter-system operability must therefore be scheduled to ensure connectivity, functionality and performance. For client server systems, separate servers will be required to support testing. Server capacity will be dependent on the same issues outlined previously for any test environment, but multiple servers will be required if applications run across multiple servers and/or different hardware vendors are utilized.

Clearly, because of the sheer magnitude of the data and programs involved, testing requirements to support the conversion effort will be extensive and impose significant computing resource constraints in the organization. The requirement for testing exists regardless of whether a manual or automated conversion methodology is employed.

A separate isolated test system will be required to support:

1. Volume of code to be changed. Given this level of testing and the sheer volume of programs that might be affected, it is easy to see why there needs to be a strong concern for the resource demands that will be made by the testing environment over the next few years.
2. Different testing environments. The testing environments will be necessary to support the following conditions: Unit testing, regression testing, integration testing, system testing and user acceptance testing.

3. System Clock Changes. A separate segregated test environment is recommended to keep production systems from contamination due to system clock advancement or untested imperfect, modified code. Extreme caution must be exercised before resetting the system timer since some system resources and functions are time-sensitive and may be activated or de-activated when you reset the system clock. These effects can occur when the system clock is set forward or backward. Careful planning will minimize the risk of system and functional losses that could be difficult and/or time-consuming to recover.

Sizing the test environment is a fairly complex task. During the early phases of testing you will be running a combination of updated in-house code and vendor supplied products. These vendor products will be used for executable manipulation, debugging and script playback. In addition, you may find a need to perform interface simulation. The good news is that these activities are not the driving force of resource requirements. The driver is the testing required to exercise large contiguous segments of code. For example, the entire nighttime batch jobs that accesses the same databases. Since it is likely that this entire segment of work will be converted at the same time to facilitate the synchronization of code and data, full production simulation tests need to be done. These tests can range from small subsets of the database to mirrors of the production environment. The large-scale integration and system tests will be the largest component of the testing process and should be used to determine the high watermark for the test configuration.

Sizing the testing platform is simplified by maintaining resource data throughout the early testing phases. During this period, the planner can capture information on how resource requirements for modified code compares to historical data from the current production versions of this same application. Depending on the amount of coding additions and the use of methods such as windowing, this difference can be very little or range into orders of magnitude.

The test plan should identify the relative size of the test database (only a subset of the production database will be necessary), the jobs that will be tested, their dependencies and the target window for completion of the test activity. Based on current resource usage of these jobs and the estimation factors (resulting from scaled down database size and code modifications), the planner can develop a sizing estimate for the test system at various stages of the project.

The planner is again faced with configuring a system that has sufficient CPU and storage resources to support testing. Data storage requirements will be based on an understanding of the relative size of test databases to current production databases, the conversion factor between the two, and the concurrency of tasks that will be tested. Given this information, the planner can formulate an estimate of data storage requirements for different phases of the testing life cycle.

A similar technique can be utilized to determine CPU requirements. In this case the inputs would be the tasks to be tested concurrently, the processor ratio between current and Y2K code and the predecessor dependencies of the tasks being tested and execution service levels. Given this information, estimates of computing resource requirements can be developed.

Estimates of testing environments equaling a quarter or more of your existing production environment are not unusual. This is particularly true as we approach January 1, 2000 and planners
need to revisit the time left versus the quantity of applications remaining to be tested. A slippage in original plans can lead to requirements for more concurrent testing and therefore, increased test capacity requirements.

It may be necessary to schedule Y2K testing during off-peak hours to accommodate the requirements but this too could impact current production batch timelines. Adequate test systems must be made available, because poor response times and job turnaround time cannot be tolerated when there is an immovable deadline.

4.3 Production impact as applications are migrated to Y2K

As applications are migrated to Y2K the impact will be felt in:
- CPU
- DASD
- Batch turnaround time
- Job reruns and outages

Sufficient capacity needs to be made available to support the migration of Y2K compliant systems to production. We expect longer execution times since modified code will require additional logic and therefore processor cycles for execution. The increased complexity of code to support 4-digit dates will require increased CPU capacity and longer online execution times and batch turnaround times. The increase to 4-digit dates can also impact file structures and block sizes. It may therefore be necessary to reorganize datasets and reexamine dataset placement across disk devices. Job reruns may be experienced due to changing code and errors inadvertently introduced by the sheer volume of applications and code that will be continually migrating to production. Job reruns and outages can also impact job schedules and critical deadlines.

It is therefore necessary to plan for additional production capacity to address these contingencies. A 20% increase in executable code can be experienced when changing the applications and at least 50% of your application portfolio is subject to change. Therefore to account for outages and additional CPU requirements, a 20% increase in processor requirements is advisable. Plan for an overall 20% increase but your requirements can exceed this, depending upon the complexity of the solution employed and the degree of date/time calculations performed particularly for financial and banking applications. Again this capacity is required now and will still be necessary even after the Year 2000 because all changes will not be finalized at the stroke of midnight on December 31, 1999.

The ability to size the production environment prior to implementation will be greatly facilitated by the collection of utilization and elapsed time data in the development and testing environments. Since the applications being implemented have, by necessity, either gone through the re-engineering or are staying in production in their native form (no Y2K changes necessary), there should be ample data to develop production utilization and delivery estimates. If the testing process was performed on subsets of the data, then extrapolation will be required to develop composite CPU requirements as well as elapsed time determinations.

At the very least, the planner can use the estimation factors that were previously developed to map current production activity against anticipated production requirements after implementation. It is important to develop the estimation factors for CPU utilization, DASD requirements and execution time, as all of these will be critical in sizing the production platform.

Data gathered during the development and testing processes will supply sufficient information to create an estimation factor for configuration
planning. Using these estimation factors and resource data from current production runs of the application, we can develop very accurate predictions of CPU and DASD resource needs.

This leaves us with the requirement of estimating the ability of the configuration to comply with production service levels with regard to execution windows. A good technique for testing the ability of the system to complete segments of work within the available window is to utilize MS Project as an estimator. All of the component jobs are input into Project with estimated run times, predecessor relationships, and required due out times. As data are collected, the individual job times can be modified and the delivery service levels can be anticipated. An example of this technique will be included in the presentation. Corporations that are employing techniques such as windowing or bridge code should be aware that there will be both CPU and elapsed time increase as a result. Bridge programs can be applied during program execution or file and/or database conversion. During program execution, the conversion occurs each time data is passed between programs or between program and source data using different record formats. During file and/or database conversion, the bridge program reads one record at a time from the source, transparently converts the record format, and writes out the data in the new format to the destination. The process is incremental and can continue until all the records in the source are converted.

The major problem with using bridge programs in actual program execution mode is that it imposes additional CPU requirements on the existing application that may impact capacity and execution times of critical systems. Essentially, the cost for every read and write of date related information will now have additional logic and program linkage associated with it.

### 4.4 Application Tuning Efforts

As you perform the integration and inter-system testing, performance optimization opportunities will be identified and it will be necessary to address these in a timely fashion so that changes can be implemented and additional testing be performed. Unfortunately there will be little time to implement and test out these changes as we approach the eve of the new millennium. These efforts will most likely be postponed until after January 1, 2000 when resources and time are less constrained.

### 4.5 Capacity Requirements After January 1, 2000.

The performance and capacity requirements will not magically disappear after January 1, 2000. You should have better estimates as to the total impact of the Year 2000 capacity requirements and can readdress capacity plans and operations schedules. But there will still be more changes to come since not all changes required to achieve Y2K date format compliance could be made as planned and temporary workaround solutions will have to be converted to permanent ones. There will be bridges and pseudo conversion efforts that must be changed and new performance tuning efforts begun. In addition, capacity will be required to support the rerunning of job failures for applications that were hastily converted in the eleventh hour. It is expected that the fallout from Y2K will last well beyond 2005 with the need to remove temporary solutions, performance tune systems and handle problems resulting from quick conversions. In addition, there will be a backlog of system plans, new applications and changes that were put off until after the Year 2000 that will require planning, coding and testing.

The additional capacity requirements should for the most part remain to support the production system. The testing requirements will be dramatically reduced and there will no longer be a
requirement to support a totally isolated and segregated test system since the “system clock” changes were the primary force behind this requirement. But there will also be new applications requiring test resources and a need to revisit all plans.

After the Year 2000, you should have a better understanding of your applications portfolio and their resource demands, systems requirements, service level requirements and inter-system dependencies. There should be better system documentation and procedure guidelines for handling problems and outages.

5.0 Summary

While the Year 2000 problem seems almost trivial at first glance, it poses the greatest challenge in the history of data processing. A major amount of concern has been focused on the application changes that are required, but of equal import to the enterprise are the exposures in the areas of resource capacity and systems performance. It is imperative that these issues be addressed throughout the Y2K effort even if the conversion may prove successful, the implementation of the solution may cause significant operational outages in the data center. Resulting capacity and performance problems could conceivably lead to the inability to service the basic tactical needs of the organization.

With limited time and resources remaining, corporations need to reassess their ability to be fully Y2K compliant. As a result of this reevaluation, major changes to resource requirements both prior to and post January 1, 2000 may be necessary. The capacity planner must play an integral role in the re-evaluation process to insure that adequate resources are in place to meet these new, more aggressive timelines.

6.0 References