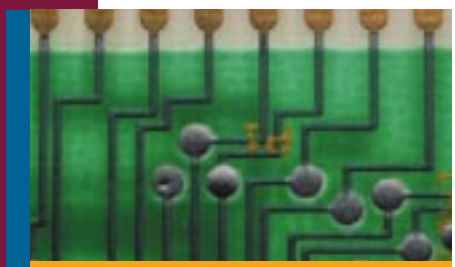


Year 2000 Testing: The Non-Technical Challenge

BY PAUL KELLY

In a number of Year 2000 projects, the focus has moved very quickly from the technical challenges of finding dates, creating isolated test environments and using tools to a primary focus on managing the scope of the project. Scope management is essentially a non-technical challenge, although technical input is required.

WHAT should I be testing for Year 2000 compliance and how much should I test it? If you haven't asked yourself these questions yet, then you can be sure that somebody else will, and soon.

In most organizations the Year 2000 project is reaching the crucial phase where testing is now firmly on the critical path. (If it isn't, then maybe it's time to put Year 2000 at the top of the list of issues that you raise with your senior managers.) The project's focus is not just how to perform Year 2000 testing but when to stop!

In other words, how much Year 2000 testing is enough? Or, perhaps that should be how little Year 2000 testing can I get away with — right?

If your responsibilities are for the infrastructure that supports the delivery of applications and services to your organization's business community, then you are likely to have a portfolio of systems and system management tools. Some will be provided by third parties; others will have been developed in-house using a variety of different tools and languages. They will reside on a variety of different hardware and operating system platforms. Potentially, you have a massive testing requirement for Year 2000.

In a number of Year 2000 projects, the focus has moved very quickly from the technical challenges of finding dates, creating isolated test environments and using tools to a primary focus on managing the scope of the project. Scope management is essentially a non-technical challenge, although technical input is required. The following steps have been developed to help put this challenge into perspective:

Step 1: Inventory — Understanding the Potential Scope

Step 1 requires that you have a complete and up-to-date list of all the components, products and systems within your scope. Hopefully, by now this information has already been collected and reviewed for the organization's Year 2000 inventory and assessment phase.

Step 2: Assessment and Update — Understanding the Planning Constraints

Step 2 requires that you understand the current levels of hardware and software installed, when the Year 2000-ready versions will be made available and when they are to be implemented. This information and these dates provide the constraints within which the Year 2000 test plan or schedule will have to fit. There will also be other constraints on the time-line:

- any internal Year 2000 project target dates set
- important calendar or business dates (in addition to 12/31/1999 and 1/1/2000)
- the availability of suitable test environments and test resources
- any dependencies between different components and their respective upgrade paths
- today's date

Step 3: Test Policy — Determining the Amount of Testing Required

Step 3 requires that a risk assessment for all of the systems and components within

your scope be performed. To explain why this is necessary, we first need to ask the question "Why do we test?" The answer of course should be obvious: We test to find errors. When we find errors in testing we should be pleased, unless of course the test plan does not provide any time allowance for resolving and retesting the errors found. However, generally speaking, finding errors in testing is considered to be a good thing because the impact of an error found in testing is significantly less than if the error were to occur in production. The easiest and most powerful way to measure the impact of an error is to record the impact in dollars. The cost may be an internal charge or fine levied against your department for failure to deliver a service; more probably, it is the actual cost to the organization in terms of lost revenue and the total cost of recovering and then rectifying the error.

What should I be testing for Year 2000 compliance and how much should I test it? If you haven't asked yourself these questions yet, then you can be sure that somebody else will, and soon.

Does this help to determine the amount of testing? Well not completely, but it does provide one of the most important factors to consider, namely the potential impact of a problem.

What else should be considered? The other key factor to assess is the chance or likelihood that the error(s) will occur. By assessing the impact of an error together with the chance that the error will occur, a risk assessment is performed. This risk assessment then forms the basis for answering the questions of what to test and how much to test.

What does this mean in practical terms? First of all, you need to understand the business processes and systems that the infrastructure supports. This in itself is a

The Non-Technical Approach

The following examples demonstrate how scope management has been used to determine the level of Year 2000 testing necessary.

Example 1

A data center has already upgraded to the Year 2000-ready version of CICS, and within the Year 2000 test LPAR, application teams are testing their systems using the upgraded version. The systems inventory, however, has identified that there are three routines programmed in REXX that are used to recover backed up CICS and application files in the event of a total system crash. An assessment of these programs indicates that they are highly date-sensitive with different files recovered and different modules invoked depending upon the day of the week and date within the month.

While the routines are required very infrequently, the overall likelihood of a system crash in the Year 2000 is assessed as being higher than normal. In this circumstance, these routines would be essential in ensuring both system and data integrity. The impact of error in these routines would be high, the likelihood that these routines will be used is higher than normal, and because of the date processing within them, there is also a higher probability that they contain Year 2000 bugs.

The resultant decision was to focus testing effort on these routines, and test scripts and plans were produced to ensure that they were rigorously tested on a list of key dates including the first five business days in the Year 2000.

Example 2

A data center uses two products for the control and management of change records and problem records. One of the products is from a reputable supplier and has been recently installed. The supplier has responded promptly to any questions related to Year 2000 compliance. The relationship with the other supplier has been less than satisfactory on Year 2000 compliance with either no response or evasive responses to any questions asked. Both products have been assessed as having minimal date processing, but the two products do interface with each other and have been assessed as being critical to the management of the data center.

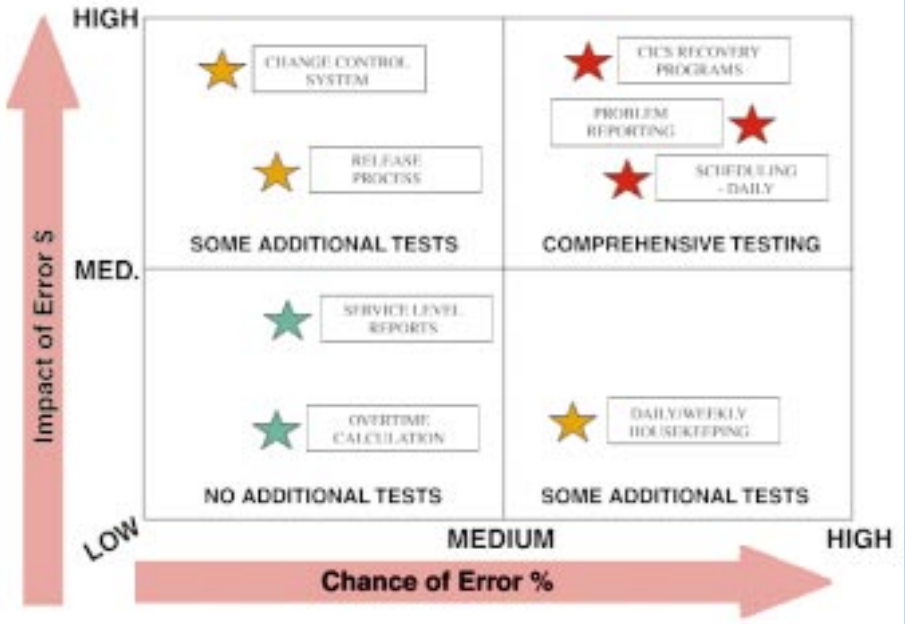
In this example the decision was to test only a small sample of the functionality in the first product but to test more fully the second product. The interface between the two products was also to be tested to ensure that even if the products were Year 2000 compliant that they had been made compliant in a complementary way.

Example 3

The data center manager uses a large number of Excel spreadsheets to calculate and report on service levels, system outages and overtime payments. These spreadsheets have been included in the systems inventory and an initial assessment has indicated that they are highly date-sensitive. Some fixes have been made to the macros and data, however, some of the spreadsheets are so old that the developers are no longer around and there is no documentation available. While important to the department, they do not immediately impact service delivery.

This example shows just how difficult the risk assessment process can be. How important are the reports, or how important is it to get overtime payments right? In this particular example, the decision was made not to test. In part this was due to the unavailability of resources to perform the testing. Had there been resources, who knows? What is clear is that faced with a potentially huge testing scope, the decisions required on what to test and what not to test will be tough but necessary.

Figure 1: Assessing the Potential Monetary Impact and the Potential for Error



worthwhile exercise if you don't already know. Once you find out, you'll be better equipped to answer the ultimate test of your worth to an organization. You will be able to answer with pride, " I support the delivery of XYZ services to our customers," when asked, "What do you do then?"

Having understood what processes or systems you support, the next key question to ask is what happens when they fail? What costs or losses are incurred? This information may in part be available in the Service Level Agreements for the systems or services; alternatively there may be people elsewhere in the department (business analysts) who are able to provide this type of information. You could, of course, speak to the service owners or business representatives in person. When given access to the right level of people, this is generally the most accurate and personally rewarding way of collecting this type of information. Once you have this information, you will now be able to answer the question, "What do you do then?" with the answer, "I support the delivery of XYZ services to our customers and if it goes wrong, it costs $ABC!"

Assessing the chance of error involves gathering answers to a lot of different types of questions. Here are a few sample questions you might ask your suppliers and yourself.

If the system, function or product is provided by a third party:

- Do you know where and how much date processing is performed in the application?
- Do you know what Year 2000 testing has been performed?
- Will the supplier share with you their test plans, scripts and results?
- When did the supplier make available the Year 2000-ready version?
- How many other sites use the product?
- How much customization have you made to the product's configuration and integration?

If the system or function has been written in-house:

- Do you know where and how much date processing is performed?
- How were the Year 2000 changes identified and made?
- Who made the changes, and how much testing did they do?

The answers to each of these questions provide qualitative and sometimes quantitative information on the chance of error.

Step 3, therefore, involves assessing the potential monetary impact and the potential for error. It provides test coverage requirements, test priorities and most importantly, the test budget (or at least an indication of it).

Figure 1 provides an example of how this information might be recorded and used. Most commonly this information is translated into a list of key processes to test, a list of key testing dates and a priority sequence. This crucial information can and should be documented. This may prove invaluable for both internal and external audit requirements as well as for the dreaded but hopefully unlikely scenario of litigation.

The Year 2000 also creates opportunities to create multi-disciplined project teams, to break down traditional demarcation lines between departments and to change traditional approaches and methods.

Step 4: Test Strategy — Designing a Cost-Effective Test System

This article does not review the different options available or the technical issues of performing Year 2000 testing. However, I would like to provide you with a few important thoughts.

The tools, processes and environments used should all be configured and designed to achieve a single objective; namely to find as many Year 2000 problems as possible in a cost-effective manner. The costs of testing should always be evaluated against the amount of testing required (see Step 3). Indeed, there will be a need to perform a "trade-off" between the desired level of testing and the costs of achieving it; this may be particularly true where the costs of test environments or capacity become prohibitively high.

CONCLUSION

In summary, therefore, the non-technical challenge facing us all is how to manage the scope of Year 2000 testing. The decisions required will be tough, and should be documented for everyone's benefit and protection. The four steps discussed here do require some technical input, but more importantly they require non-technical

decisions based on impact, chance, risk and cost.

I am often told, "I don't have time to do all this risk assessment," or "My team doesn't have access to business people or business analysts, and besides, they will be too busy to answer our questions." It is true that the processes detailed here do present new challenges for those organizations that have not already performed this kind of analysis, but, what are the alternatives? You could do no Year 2000 testing. You could waste time and effort testing non-critical systems or systems that are not significantly threatened by the Year 2000 problem. Or, you could start testing at the letter "A" in your inventory and run out of time and money by the time you reach the letter "C".

In other words, the alternative to not prioritizing Year 2000 testing based on a formal risk assessment will result in either serious problems in the Year 2000 or massive budget overruns prior to the Year 2000.

One final thought: The Year 2000 is so often presented as a problem that needs to be overcome, and indeed, a Year 2000 project has many challenges. This perception is reflected in the way that the projects are managed and the approaches adopted. However, the Year 2000 also creates opportunities to create multi-disciplined project teams, to break down traditional demarcation lines between departments and to change traditional approaches and methods. Implementing a risk-based approach to Year 2000 testing should be seen as an opportunity with significant benefits for not only your Year 2000 project but also for future projects as well. ts

Paul Kelly is a Year 2000 testing consultant working for Unisys Corporation. He is currently working with many organizations in Europe. His background includes development and operational management of MVS mainframe systems.