Year 2000 Readiness Checklists

Watts S. Humphrey
Software Engineering Institute, Carnegie Mellon University

The checklists in this article are designed to help organizations determine their readiness for the year 2000 (Y2K) date change. The lists are brief but include the items many experienced professionals have concluded are necessary for adequate Y2K preparation. They will help managers quickly assess their status and determine where they are exposed. This article also includes explanations of the various items and some general comments.

Recently, I was invited to a meeting in Washington D.C. to review a proposed quality standard for the year 2000. Although the meeting was focused on testing, the discussion reinforced the multifaceted nature of the Y2K problem. Since few people have seen a problem like this, we need to characterize it in a way that clearly identifies the actions that must be taken, and a checklist could be what is needed. It would describe the actions required and enable managers to quickly see the work they need to get under way.

After drafting the Y2K readiness checklist, I had several knowledgeable people review it. Although no brief checklist can be complete, this one covers the points the reviewers and I felt were most important. Organizations may identify additional important areas, but they should not substantially expand the checklist because that would make it harder to use and less effective.

This checklist is designed to help you judge the readiness of your organization for the Y2K transition. In using this checklist, you should consider several points.

- The Y2K transition problem will be much like those you have experienced in installing and converting to a new system or application program version, only this time, you will be installing and converting to an updated version of every application and system simultaneously.
- If you experience Y2K problems with any commercially supplied software package, other users will likely have similar problems at the same time. Thus, the vendors' help desks and support services will probably be swamped with calls and unavailable for extended periods.
- Unless you have contracted for dedicated support services, you must be prepared to be self-sufficient. If you have a support contract, you need to ensure the suppliers of such services have resources dedicated to your needs.
- The Y2K cutover will most likely result in multiple application and system crashes and other disasters. It is therefore essential that you maintain complete backups of all application and system data and programs. Note, however, that with the Y2K problem, traditional backup practices will not work. Usually, when backing up, one returns the system to a prior configuration that worked. In this case, at least until after the cutover period, there will be no prior configuration that is known to work.
- Be careful about vendor selection because there will likely be unscrupulous providers of Y2K services. When organizations do not have a competent technical staff, dishonest operators could pretend to do the required work, then disappear before Jan. 1, 2000. Any organization that wants substantial payments upfront should be avoided like the plague. Although scams are nothing new, Y2K is an extremely attractive opportunity; any organization that is victimized could well be out of business before it can recover damages.
- View all Y2K tools, services, guidelines, and checklists (including this one) with a high degree of skepticism. Remember, none of them have been tested in practice and shown to work. You therefore must examine all such offerings and satisfy yourself that they are credible and reliable.
- Consider each specific item in this checklist and satisfy yourself that it is necessary for your organization. With the exception of the crisis response, zero-hour testing, and emergency backup, all other items except one may or may not be essential, depending on your situation. Although you may not need these essential items, if you do (and most organizations will) you cannot build them on short notice.
- The one capability exception that every organization should put in place as soon as possible is a configuration management system. If you do not have this capability, you will most likely have problems that could be severe and unrecoverable. This must not be viewed as something to do later when you have a chance. Put a configuration management system in place now.

The Y2K Readiness Checklists

The following checklists are designed to help you quickly assess the readiness of your organization for the Y2K transition. Complete the organization checklist first (Figure 1), then fill out the application checklist for every active application (Figure 2).

This article is based on "What Does Y2K Mean to You," Object Magazine Online, April 1998 (http://www.sigs.com/omo).

Completing the Checklist

Have the most knowledgeable engineers and managers complete the checklists. Brief descriptions of the checklist terms follow the checklists. In completing the checklist, the columns to the right refer to the status of the application or the overall organization. If all the required work has been done, check the "done" column. Similarly, if the work is not yet done but the project is staffed and under way, check the "under way" column. Later, when the application checklists are completed, you can enter percentage values for the percent of applications that have been completed for each checklist category.

Figure 1. *The Y2K organization readiness checklist.*

*	Organization Units (laboratory, plant, etc.):	Done	Under Way	Plan	No Plan
	Should be done by Jan. 1, 1998.				
1	Adequate 1998 budget and staff in place.				
2	Configuration management system in place.				
3	Applications inventoried.				
4	Applications assessed (with checklist).				
5	Application source code under control.				
6	Application priorities determined.				
7	Y2K tools available.				
8	Date change standards defined and tested.				
9	Database correction inventoried.				
10	Service and support facilities surveyed for Y2K.				
11	Critical applications staffed and in development.				
	Should be done by Jan. 1, 1999.				
12	Adequate 1999 budget and staff in place.				
13	Applications updated and in test—*critical*.				
14	All applications staffed and in development.				
15	Database corrections staffed and in development.				
16	Service and support corrections under way.				
17	Hot-line group funded and staffing identified.				
	Should be done by July 1, 1999.				
18	Backup procedures defined.				
19	Applications updated and in test—*key*.				
20	Emergency procedures defined and tested.				
21	Customer and supplier testing under way.				
22	Zero-hour procedures defined.				
23	Hot-line groups staffed and supporting testing.				
24	Database corrections in test.				
	Should be done by Sept. 1, 1999.				
25	Emergency procedures training in place.				
26	Applications updated and in test—*all*.				
27	Zero-hour procedures tested.				
28	Backup procedures tested.				
29	Service and support tests completed.				
	Should be done by Dec. 1, 1999.				
30	All application and database testing complete.				
31	Adequate Y2K budget and staff in place.				
32	Customer and supplier testing completed.				
33	Emergency procedures training complete.				
34	Backup procedures in full operation.				
35	Zero-hour testing staffed for Jan. 1, 2000.				
36	Hot lines fully staffed and rehearsing procedures.				
	Should be done by Jan. 1, 2000.				
37	Emergency procedures rehearsals completed.				
38	Zero-hour testing under way for Jan. 1, 2000.				
39	Zero-hour testing staffed for Feb. 19, 2000.				

The Y2K Application Checklist

Complete the application checklist for every active application in the organization.

Checklist Definitions – Checklist Columns

The four columns to the right of the checklist are for status information. In most cases, this work is either done or not done. Generally, it is desirable to either check the item or enter a date when it will be done. After an initial assessment, and after the work is under way, it is helpful to enter a percentage figure, as with "Applications assessed (with checklist)." Here you would enter the percentage of the applications assessed. Note, however, that the columns should add to 100 percent when using percentage values for a checklist row.

It also is important to only count completed work. For example, if you have 10 items and three of them are finished, that would be 30 percent complete. When six of 10 items are each half done, however, you would have zero percent done. Do not take credit for partially completed work because partial status is generally hard to estimate and can be misleading.

The status levels are defined as follows:

Done – This column is checked when the listed work has been completed.

Under Way – This column is for work that is staffed and under way. In those cases where the work is only 50 percent staffed, for example, it would be more informative to enter a 50 percent in this column. Note that this column does not give any indication of whether the work is likely to be completed on schedule.

Plan – This column is for those cases where the work is planned but it is not yet staffed or the staff may have been identified but the work has not yet started.

No Plan – This column is for those areas where the work has not yet been planned

or it has been planned but no staff has yet been identified to do the work.

***** – Where any tasks are late or need special attention, mark them with an asterisk in the "*****" column.

Application Priorities

The definition of critical applications is a matter of judgment and must usually be settled by senior management. It is therefore suggested that a comprehensive listing of all applications be reviewed with management together with a list of those applications that are deemed critical and why. This is the action called for under "Application priorities determined." These priorities should specify which programs are critical, key, active, and inactive and also which will need emergency backup procedures, hot-line support, or zero-hour testing.

The application priorities are as follows:

Critical – These are the applications on which the organization's business depends. They can be found by identifying the work that would have to be done manually if all computers were shut down for weeks or months. Without the critical applications, the business could not function.

Key – The key applications are those the business needs but are not a matter of organizational survival. Although they are needed, their unavailability for periods of weeks and even a few months would not be fatal; that is, either the application work can be deferred or manual back-up procedures will be devised to handle the needs in the interim.

Active – The active applications are those actively in use other than critical or key applications. These applications may only be run once a year or occasionally on demand. Although they are needed, their repair can be deferred until shortly before the application is needed. Note, however, that some applications may only be used once a year for tax purposes. If that one-time use is in January, however, this could become a critical application.

Inactive – These are all the applications that are no longer in active use. In most cases, these would not warrant a Y2K repair effort.

Readiness Dates

The dates given in the checklist are selected based on overall judgment of the amount of work to be done in a small- to medium-sized organization that has a competent software staff on hand. These are the latest advisable dates; organizations should strive to get this work done earlier if possible. Also, if the organization is extremely large or if it does not have a reasonably large and competent staff, these dates should be even earlier.

For large organizations, earlier dates are needed because of the huge volume of work. Smaller organizations without a substantial information systems staff will need to hire one or more suppliers of Y2K services. It takes time to identify and obtain these services, so these organizations should work to meet the earliest possible dates.

Alphabetical Glossary

Following is an alphabetical listing of definitions of many of the items on the readiness checklists. The topic headings are the same as in the checklists, and the numbers refer to the numbers in the asterisk column.

(1, 12, 31) Adequate (year) budget and staff in place – This should be management's top priority. Unless the

Figure 2. *The Y2K application checklist.*

	Application Name:				
	Application Priority:				
	Application Support Needs			Yes	No
	Emergency procedures required.				
	Hot-line capability required.				
	Zero-hour testing required.				
*		Done	Under Way	Plan	No Plan
	Should be done by Jan. 1, 1998.				
50	Application priority determined.				
51	Source code available.				
52	Source code checked against object code in use.				
53	Critical applications: staffed and in development.				
54	Applications under configuration management.				
55	Database correction inventoried.				
	Should be done by Jan. 1, 1999.				
56	Applications updated and in test—*critical*.				
57	All active applications: staffed and in development.				
58	Database corrections staffed and under way.				
59	Hot-line groups funded and staffing identified.				
	Should be done by July 1, 1999.				
60	Emergency procedures defined and tested.				
61	Applications updated and in test—*key*.				
62	Critical applications tested with updated database.				
63	Hot line partially staffed and supporting testing.				
	Should be done by Sept. 1, 1999.				
64	Emergency procedures training in place.				
65	All applications updated and in test.				
	Should be done by Dec. 1, 1999.				
66	All application testing complete.				
67	Emergency procedures training complete.				
68	Hot-line fully staffed and rehearsing procedures.				
	Should be done by Jan. 1, 2000.				
69	Emergency procedures rehearsals completed.				
70	Zero-hour testing under way.				

work is staffed in time, there is no way to finish in time. The adequacy of the funding and staffing should be based on an assessment of a plan to do the work and an estimate of the resources required.

(4) Applications assessed (with checklist) – This refers to applications being assessed with the application checklist.

(3) Applications inventoried – Every application in use must be identified. This requires naming the application and where and when it runs. The inventory also should list available source code, manuals, procedures, and guidelines about the application, who uses it, and when. This information is needed to set priorities.

(6, 50) Application priority determined – Management must decide which applications to fix, which to replace, and which to handle with a hot-line capability. This sets the priorities for every application and guides the allocation of development work. If these decisions are not made early in the Y2K program, important programs will likely be overlooked while less critical applications are being fixed. Management must set priorities at the earliest possible point: which applications are critical, key, active, and inactive.

(5) Application source code under control – see "Configuration management" and "Source code available."

(54) Applications under configuration management – see "Configuration management."

(13, 19, 26, 56, 61, 65) Applications updated and in test – once corrected, the applications must be tested with the corrected databases. It is important that this testing cover the applications' functions as well as all the code and database changes.

(18, 28, 34) Backup procedures – The Y2K cutover will most likely result in multiple application and system crashes and other disasters. It is essential that complete backups be maintained of all application and system data and programs and that these backups be updated frequently. All old backups must also be retained since files can be unknowingly corrupted and not discovered until much later. Good practice dictates that backups be taken as early as possible, even before Y2K work starts. Note, however, that with Y2K, traditional backup practices will not work. Usually, when backing up, one returns the system to a prior configuration that worked. With Y2K, at least until after the cutover period, there will be no prior configuration that is known to work.

(2, 54) Configuration management – The configuration management system maintains physical and electronic control of the organization's programs and data. Applications that have been in use for many years often have not been changed for much of that time. Unless the organization has an established configuration management system, the source code could have been lost. The configuration management system is also needed to ensure that the changed programs are properly updated in test and that only tested programs are put into use. Without an effective configuration management system, organizations are likely to lose programs, misapply fixes, or use the wrong tests and test data. All this wastes time, which is the one thing you cannot recover.

(21, 32) Customer and supplier testing under way – Many businesses have critical dependencies on their customers or suppliers. Where these relationships involve data processing interactions, there will likely be Y2K problems. The fixes to these problems must be tested in advance.

(9, 15, 24, 55, 58) Database corrections – Depending on the Y2K change strategy, many database changes may have to be made. Also, during the transition, there are many ways that databases can be corrupted. Until programs have been corrected, for example, many date calculations will put incorrect dates in the database. After the programs are corrected, subsequent dates will be correctly calculated. The application, however, will not go back and search for the incorrect entries in the old data. This must be done by hand or with special tools, if any can be found. This issue is further complicated by the phased cutover of multiple applications. The corrected databases must then be tested with the corrected applications.

(8) Date change standards defined and tested – The Y2K date conversion formulas are not complex, but they are not trivial. It is essential that the engineering change teams know precisely how to handle date calculations.

(20, 25, 33, 37, 60, 64, 67, 69) Emergency procedures defined – With large systems and large volumes of changes, there will be many defects. Thus, even the most critical applications will likely be unavailable for periods. The organization must be prepared for this eventuality and have a capability in place to handle any problems. The emergency procedures define how an application is handled under these conditions. Manual procedures should be in place and tested for all critical and selected key applications. Because large numbers of people will need to know how to quickly respond in an emergency, training programs and procedure rehearsals will generally be needed.

(17, 23, 36, 59, 63, 68) Hot lines – The hot-line support group handles the crisis calls when applications fail during and after the Y2K cutover. Since application failures can occur early for applications with advanced dates (like credit card expirations), the hot-line groups may have to be staffed much earlier, depending on application needs. It also is important to staff these groups early to give them experience with the applications they will handle. The best way to do this is to have them in place handling Y2K testing problems and fixes.

A hot-line support capability will be needed even for those applications that have been completely repaired and tested, because between 1 percent to 20 percent or more of all the Y2K fixes will

likely have problems, even after testing. With a fix quality program in place, the 1 percent number is achievable. If not, 20 percent or more is likely, depending on staff experience, program complexity, and the degree of testing.

Generally, you will need application-knowledgeable people to staff the hot lines. They provide telephone assistance to system and application users, internal or external. Their job is to help the application users when they run into problems.

(10, 16, 29) Service and support – Many facilities such as elevators, security systems, power distribution, telephone systems, and air conditioning could have date dependencies. Although these items may not have problems, they could. Each one should be tested or checked with the manufacturer for Y2K compliance and warranty coverage.

(51) Source code available – If the source code for any active program is not available and the program has date dependencies, the application must be replaced and tested. Replacement may be expensive and take time, but without the original author, programs can rarely be fixed without the source code.

(52) Source code checked against object code in use – In older systems, applications were occasionally patched to avoid the time required to recompile and rebuild. When this has been done, the source code will not represent the program that is being used. When the developers update the source code for the Y2K fixes, compiling and installing it will erase all these prior object corrections. The result will be the simultaneous re-emergence of all the problems the original object patches were designed to fix. These problems will not wait until the year 2000; they will happen starting now. To resolve such problems, these patches must be identified and added to the Y2K workload. These patches can be found by compiling a new object program from the source and making a bit-for-bit comparison with a copy of the object program in use.

(7) Y2K tools available – This box should only be checked after the tools have been evaluated, obtained, and tested in practice. Until they are, it is likely that many of the tools will be found less effective or harder to use than promised.

(22, 27, 35, 38, 70) Zero hour – The zero hour is midnight Friday, Dec. 31, 1999. Over the long weekend of Jan. 1, 2000, the large and complex Y2K system change must be tested live for the first time. The zero-hour procedures define how the time between Thursday, Dec. 30, 1999 and Tuesday, Jan. 4, 2000 is to be used. During this period, many groups should make test application runs both in-house, with suppliers, and with customers. This weekend is also when the hot-line support groups must move up to full capacity. The zero-hour procedures are used to phase repaired applications into service and recover from any problems found. Although this cutover should be started as early as possible, special recovery resources and procedures must be available and tested during zero-hour testing. Plan to maintain the zero-hour testing capability until all the critical applications are cut over and work properly. This will likely take at least a month and could take much longer. Some organizations plan to maintain this special testing and support capability for at least three months.

(39) Zero-hour testing under way for Feb. 29, 2000 – Normally, there is a leap year every four years. The exception is every 100 years when there is not a leap year. This too has an exception every 400 years when there is a leap year. Thus, 2000 is a leap year, and there will be a Feb. 29, 2000. It is important that the date change algorithms be clearly defined and disseminated so everybody working on this problem understands them. Since these date algorithms will first be tested Feb. 29, 2000, a zero-hour testing plan is needed.

Summary

Any organization that does not have an active Y2K program under way had best get started immediately. The date when the work will be completed is principally determined by when the work starts. If you are still studying, stop and get to work. Make a plan at the same time, but get to work! ◆

About the Author

Watts S. Humphrey is a fellow at the Software Engineering Institute (SEI) of Carnegie Mellon University, which he joined in 1986. At the SEI, he established the Process Program, led initial development of the Capability Maturity Model, introduced the concepts of Software Process Assessment and Software Capability Evaluation, and most recently, the Personal Software Process and Team Software Process. Prior to joining the SEI, he spent 27 years with IBM in various technical executive positions, including management of all IBM commercial software development and director of programming quality and process. He has master's degrees in physics from the Illinois Institute of Technology and in business administration from the University of Chicago. He is the 1993 recipient of the American Institute of Aeronautics and Astronautics Software Engineering Award. His most recent books include *Managing the Software Process* (1989), *A Discipline for Software Engineering* (1995), *Managing Technical People* (1996), and *Introduction to the Personal Software Process* (1997).

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Voice: 412-268-6379
E-mail: watts@sei.cmu.edu