A Y2K Integration Test Model

Dr. William H. Dashiell
National Imagery and Mapping Agency

An Integration Test Model provides Year 2000 (Y2K) integration test objectives keyed to specific integration test cases. This article describes a suggested basic year 2000 (Y2K) test model with lessons learned.

Background

The Y2K problem centers on the interpretation of a two-digit code representation of a year. Simplified, the Y2K problem may be seen as many computer software applications that have been programmed to interpret the two-digit year range of 00 ... 99 as meaning the years ranging from 1900 to 1999. Other software applications interpret 99 as an end-of-file or end-of-data marker. Regardless of how the problem is defined, one will interpret it as catastrophic when one's critical need is not addressed or addressed incorrectly — sometimes with irretrievable results.

Because of costs or lack of programmer resources to correct the source code (assuming that the source code is available and that an executable image may be generated) the basic quick fixes for the Y2K problems often cause two effects:
1. delaying the impact of the Y2K problem by using techniques to interpret various ranges of dates such as bridging, sliding, or fixing windows.
2. exchanging what is essentially either a single date or a well-defined collection of dates for multiple dates with unknown impacts.

Regardless of the “fix” used for the Y2K problem, formal Y2K testing to a set of Y2K objectives should be done to assess the level of risk to which the users of those applications are exposed.

Unit Testing vs. Integration (Interoperability) Testing

In more traditional software testing environments, unit testing of an application (e.g. software using services and facilities provided by an information system specific to the satisfaction of a set of user requirements) usually involves the testing of a module within a larger, whole software application entity. In the context of this article, unit testing of an application involves the whole of a single software application entity. The software application may perform specified function(s) within the computer system. Unit testing usually is performed on a hardware platform (e.g. a collection of hardware and software components that provides the services used by support and mission-specific software applications) with or without other software programs being visible, such as an operating system. Note that the hardware platform may include a client/server-based system or a Web-based system and their respective required software to enable each system to function as designed. Unit testing is not integration testing but is generally performed prior to integration testing.

One important objective in software testing is the validation of the application(s) under test (e.g. those applications that are subject to testing requirements). Validation testing is a process of assessing the conformance of one or more software applications to one or more standards or to a set of specifications. This process includes the administrative procedures to set up conformance assessment and to issue some formal document, such as a certificate or test report, that an agreed upon or recognized process was followed and that records which of all tests presented were passed. For tests that were failed, the formal document notes which tests failed and specifies the functionality assessed. The user of the formal report documenting failed tests may find that the functionalities represented by the tests are not needed.

Why Perform Integration Testing?

The primary purpose of testing is to satisfy a customer’s needs and requirements. Unit testing primarily assesses the validation of an application by itself. However, when multiple applications share resources, the closer the testing environment is to that of the customer's environment the more likely that testing will detect anomalies. By design, integration testing encompasses multiple applications and uses either the customer’s environment or a separate test environment that closely duplicates the customer’s
environment.

Customers
Identify the customer when designing the integration test environment and test script. For example, consider the following four categories of customer sets:

- **End-users** — should be the highest support priority customer set.
- **Operators** — should be the second priority customer because they usually are individuals who are an integral part of the production process. This customer set often operates and manages data centers.
- **Maintainers** — are hardware, software, and network infrastructure personnel who maintain the operational systems and provide on-the-floor support to the system user community.
- **Developers** — the individual product developers, such as the project managers, technology thrust managers, and capability security certification administration.

Year 2000 Integration Testing
The Y2K integration testing is designed to ensure the continuing integrity of a user's base line and to provide customers and end users with continuing integrity of that base line. While performing all integration testing, integration testers should seek to provide operational acceptance with zero open discrepancy reports (DRs). The Y2K test scripts and test reports are designed to ensure that the reported test results are accurate and repeatable.

The Y2K Integration Test Model involves its customers and end-users in the integration testing process to ensure user acceptance as well as technical base line acceptance for newly delivered capabilities.

There are five basic phases in the Y2K Integration Test Plan. While these phases were implemented in the order presented below, the awareness phase is an ongoing phase because of software changes (e.g. through application of patches, new builds, request for new functionality(ies), and results of the renovation phase) that occur throughout an application's life cycle.

Awareness
In the awareness phase, all personnel responsible for the development, testing, or who use the information technology (IT) system have been educated about the importance and impact of Y2K problems.

Assessment
This phase requires that all IT components are first unit tested by a separate unit test group. Once the unit test group successfully tests a software component, that component is transferred to the Y2K integration testers, who perform the Y2K integration test scripts on the set of software/hardware components comprising the applications under test. When applications under test are not unit tested, this Integration Test Model suggests that integration testers should perform the integration test script with the knowledge that sources of errors may not be easily traced. The assessment phase includes a strategy and plan to correct the deficiencies with full regression testing of the applications under test.

Renovation
The renovation phase documents the software/hardware changes, obsolescence of software/hardware, and upgrades to software/hardware. (Renovation is performed by other activities or organizations.) Full regression testing of renovated applications is strongly recommended.

Validation
This phase describes the test and verification process for all IT software components possibly affected by the Y2K problem. All validation testing is designed to occur in an isolated testing environment wherein regression and future software integration testing may be performed without impact on operational production systems. Full regression testing of all replaced or converted system components should be done.

Baseline
The base line phase describes the operational base line of software wherein the newly tested software is integrated. A properly constructed baseline supports multiple control levels; provides for storage and retrieval of configuration items/units; provides for the sharing and transfer of configuration items/units between control levels within the library; ensures correct creation of products from the software base line library; supports generation of reports; and, provides for the maintenance of the library structure.

Objectives
The primary objective is to ensure full regression testing of all software components for Y2K compliance.

In carrying out the integration testing responsibility, specific goals have been derived to govern the general operational testing procedures and particularly Y2K integration testing to:

- maintain a focused commitment to and support of the migration of legacy systems into a base line;
- identify and respond quickly to changing priorities;
- partner with your software system control personnel (e.g. executive decision makers) and your user community to ensure compatible, integrated test planning, scheduling, and execution to minimize the need for partial capability acceptance and retest;
- adhere to all of your software community standards, policies, and procedures;
- provide testing that ensures the continuing integrity of your operational base line;
- involve your customers and end-users to ensure user acceptance as well as technical base line acceptance for newly delivered capabilities;
Tests whether software properly processes current day, date, and time. A basis to start testing.

1.0 Tests roll-over of the Global Positioning System (GPS) 10 bit epoch. Days are correctly recognized as Saturday and Sunday, respectively. See GPS note.

2.0 The numeric value of the day (999) is equal to the null void code sometimes used in programming. Day is correctly recognized as Thursday.

3.0 Tests critical roll-over of federal fiscal year 2000 roll-over. Days are correctly recognized as Thursday and Friday, respectively.

4.0 Critical midnight crossing from 1999 into the year 2000. Days are correctly recognized as Friday and Saturday, respectively.

5.0 First day back to work for most employees Wednesday, Sept. 8, 1999

6.0 Tests roll over from single digit month to double digit years in year 2000. Day is correctly recognized as Monday.

7.0 Critical midnight crossing from 2000 into the first leap year after year 2000 begins. Days are correctly recognized as Tuesday and Wednesday, respectively.

8.0 Tests roll-over from single digit month to double digit month in year 2000. Days are correctly recognized as Saturday and Sunday, respectively.

9.0 Critical midnight crossing from 2000 into 2001. Tests roll-over to new millennium. Days are correctly recognized as Sunday and Monday, respectively. This date is the last day of the second millennium on the Gregorian calendar. The ordinal date 00.365 was the last day of 1900 (Julian Calendar). Since 2000 is a leap year, its last day is 00.366. An incomplete algorithm for determining the length of the year might cause an ordinal-based system to transition into the new millennium a day too early.

10.0 Tests roll-over from first leap year not affected by a century or millennium transition. Days are correctly recognized as Sunday and Monday, respectively. Julian date (sometimes called Ordinal Date) function should return N°th day of year.

Table 1. Y2K test script dates.

- ensure test scripts and test base lines are developed that can produce accurate and repeatable results in satisfying the test requirements;
- achieve scheduled testing deadlines established by the customer;
- proceed to operational acceptance with zero open DRs.

Scope of Y2K Integration Testing

As a first step, the integration tester is urged to test for proper processing of the current date and time prior to starting the Y2K test dates. The integration tester should be a software tester with professional experience who will review each test objective and decide its applicability to the applications under test and to modify those test objectives and test procedures to more properly match the functionality of the applications under test. This professional experience allows the tester to make professional judgements and evaluations based upon the test objective and his or her testing experiences.

The integration tester must provide an audit trail. The recommended methodology is to leave each test objective and procedure as written. In the test report, the tester should document each deviation from the objectives or procedures, with a rationale for each change.

Certain dates are widely recognized as among the most important in Y2K integration testing. These dates, which form the basis for the Y2K test script, are shown in Table 1.

Global Positioning System Note: Users of the Global Positioning System (GPS) should note that GPS does not have a Y2K problem. However, a clock overflow problem, called the “Z-count roll-over” does exist and is sometimes erroneously labeled as a Y2K problem. This clock roll-over occurs every 1,024 weeks; the first roll-over having occurred Aug. 21, 1999. Despite the publication of a GPS specification, some receiver manufacturers did not account for the Z-count roll-over in the satellite clock. Some affected receivers can be manually reset, or if they have flash memory or removable Programmable Read Only Memory (PR ROM), they can be reset to accommodate the roll-over. Those that cannot be reset must be replaced.

The following selected generic test objectives are widely recognized as the important test objectives in Y2K integration testing. It is the responsibility of the integration tester to select those objectives that are applicable to the applications under test and to develop a formal test procedure and a formal expected results for each selected test objective. The selected generic test objectives are shown in Table 2.

Sample Integration Test Script

Each Y2K test objective is developed into a specific test that the tester uses as a basis for assessing conformance to Y2K requirements. Each tester is encouraged to pursue additional testing when errors or abnormalities appear.

All testing procedures are reported in the test report with the observed test results. Below is an example of a test objective with its associated test procedure(s) and expected results.

Note for tester: When a test objective is not applicable to an applications under test, use the following statement:

Recording results: The test objectives are not applicable to the applications under test because the required functionality is not supported.

Test No. 1

Test objective (TO) No. 1: Tests roll-over of the GPS 10 bit epoch. Days are correctly recognized as Saturday and Sunday, respectively.

Test procedure, Part A for TO No. 1A: Set system date to Saturday, Aug. 21, 1999 (1999-08-21) at or about 23:00 hours. Check each commercial-off-the-shelf (COTS)/government-off-the-shelf (GOTS) application in turn for the correct date and time. Exchange the current date and time between appropriate applications and check that the date is correct within the time period.

Note to tester: Set time sufficiently prior to midnight to allow you to assess each of the applications under test in a timely manner.

Expected results: Date must be Saturday Aug. 21, 1999
between 23:00 and 23:59 hours.

Recording results: Record the result for each application as “passed,” “failed,” or “n/a.”

Test procedure, Part B for TO No. 9:
Wait long enough to allow date to roll over. Check applications for date and time and again exchange the current date and time between appropriate applications and check that the date is within the correct time period.

Expected results: Date must be Monday, Jan. 1, 2001 between 00:00 and 00:59 hours.

Integration Test Report
The Integration Test Report should provide:
• a full description of the software/hardware test environment
• a test number to identify the test report
• the test preparations (e.g. obtaining all software in a correctly configured format)
• the test script (or a reference to the formal test script to allow future replication)
• a full description of the testing procedures, including any additional testing resulting from observed abnormalities, or changes to the test objective and/or test procedure and the rationale for the changes
• the operator notes (e.g. background information, history, glossary, rational), as needed
• any acronyms used in the test report
• any points of contact (e.g. names, addresses, and telephone numbers)
• a recommendation (e.g. whether the software is approved for inclusion into the standard build-up-grade; or approval is denied with an explanation.)

Lessons Learned
• There are several COTS products that vendors claim are Y2K compliant. These products are Y2K compliant with a shift in the way end-users enter their data into the application; there are no technical workarounds. It is the responsibility of upper management to provide the basis for a policy directive to change the way end-users enter data. These known problems were not used when developing the suggested Y2K integration testing script.
Y2K integration testing is not validating the results of unit testing. A tester should review the documents associated with unit testing and may use them as a basis for the integration testing. In some instances, the tester may find omissions of, or inconsistencies in, required data in the unit test reports. In these instances, the tester should work to resolve these discrepancies because inaccurate unit test reports could invalidate the integration testing efforts.

Some applications may not coexist on the same operational system. For example, different versions of Microsoft Office will not coexist on the same testing system at the same time. Therefore, two tests must be conducted for each system. For example, integration testing should be conducted with one version of MS Office and all other applications, then with a different version of MS Office and all other applications.

About the Author

William H. Dashiell is a computer scientist at the Department of Defense National Imagery and Mapping Agency. He has worked on the development of software testing by statistical methods using binomial models, coverage designs, mutation testing, and usage models. He has contributed to the development of conformance and testing protocols for federal, national, and international information technology standards. He has a bachelor’s degree in business administration and education, a master’s degree in education technology, and a doctorate in mathematics education from the University of Maryland. He also has a master’s degree in computer science from Hood College in Maryland.

Further Readings

2. URL: http://www.nist.gov/y2k/datetest.htm (Test Assertions for Date and Time Functions).
4. URL: http://www.microsoft.com/technet/topics/year2k/default.htm (Microsoft Year 2000 Readiness Disclosure and Resource Center Website).

Call for Articles

If your experience or research has produced information that could be useful to others, CROSSTALK will get the word out. We welcome articles on all software-related topics, but are especially interested in several high-interest areas. Drawing from reader survey data, we will highlight your most requested article topics as themes for future CROSSTALK issues. In future issues, we will place a special, yet nonexclusive, focus on

- Risk Management
  February 2000
  Article Submission Deadline: Oct. 1, 1999

- Education and Training
  March 2000
  Article Submission Deadline: Nov. 3, 1999

- Cost Estimation
  April 2000
  Article Submission Deadline: Dec. 4, 1999

We will accept article submissions on all software-related topics at any time; issues will not focus exclusively on the featured theme.

Please follow the Guidelines for CROSSTALK Authors, available on the Internet at http://www.stsc.hill.af.mil.

Ogden ALC/TISE
ATTN: Heather Winward
CROSSTALK Associate Editor/Features
7278 Fourth Street
Hill AFB, UT 84056-5205

Or e-mail articles to features@stsc1.hill.af.mil. For more information, call 801-775-5555 DSN 775-5555.