Chapter 2

Theoretical Consideration

Year 2000 Problem is basically just a simple problem relate to the computer calculation as we know. However, the year 2000 problem is not as simple as it may sounds when it involves a lot of systems that tightly integrated together. To make the situation worse, we need to accomplish all of the targets before year 2000. Therefore the challenge is not the technical but the managerial. The success of the program is depends highly upon the executive leadership and program management.

In order to deal with the problem within the time limit, the year 2000 program need to treat it as a Project. With the Project Management Tools and Techniques, it will enable the organization to handle this sophisticated and customized situation systematically. Especially when it involves the large-scale software conversion or system development efforts in order to achieve the success.

In additional of the time constraint, the software testing is also requiring basic technical knowledge involve with the date calculation for Operating System, Applications, embedded memory.

Project Management & Year 2000 Software Testing

Software testing will allow the company to evaluate the performance, functionality, and integration of the converted or replaced platforms, applications, databases, utilities, and interfaces in an operational environment. However, this is a task that has clear objectives and Time Constraint. From this point of view, we can consider this as Project Management.

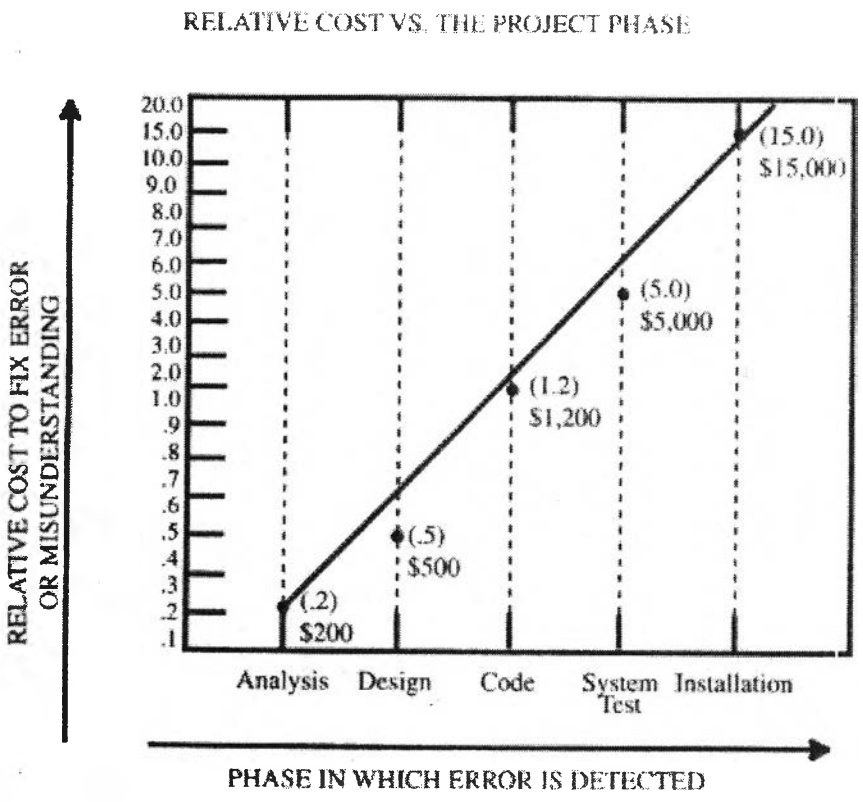
Objectives of Testing

All testing are focuses on discovering and eliminating defects or variance from what is expected. There are 2 types of defects:

- Variance from Specifications. A defect from the perspective of the builder of the product.
- Variance from what is Desired. A defect from the user point of view.

Year 2000 Testers need to identify both types of defects.

The cost of defect identification and correction increases exponentially as the project progresses. As the figure below illustrates the accepted industry standard for estimating costs, and shows how costs dramatically increase the later the testers find the defect.



Summary of IBM, GTE, and TRW Survey
Source: B. Boehm "Software Engineering"
IEEE Transactions on Computer, Dec. 1976.

Figure 2.1 : Relative cost Versus the Project Phase.

A defect encountered during requirement and design is the cheapest to fix. Therefore, it is far most cost-effective way to develop an error-free system. Testing should begin at the first phase of the life cycle and continue throughout the life cycle.

Define Risks

A Risk is a condition that can result in a loss the greater the loss will occur, the greater the risk. However, the risk exists even if the potential loss does not occur. This is due to the preventive measures that can be put in place to prevent the loss. This can be achieving by having a full understanding of the probability of the type of loss that involve. However, the fully understanding of the type of loss itself does not prevent the loss. The elimination of the root cause of the risk will reduce the loss.

There are two type of millennium failures, they are:

(1) **failures within individual components** (databases, communication standards, message busses, control systems, user screens, real-time clocks, operating systems, data files, engineering applications, monitoring software, station controllers, and so on), and

(2) **System failures caused by interactions among failing components**. Therefore testing will have to be done at both the component and system levels. It is the system-level failure mode that makes this Year 2000 problem really difficult to resolve. The problem has been likened to catching turkeys, wherein catching one turkey is not particularly difficult, but catching a whole flock of turkeys in a specified sequence is very difficult indeed.

Risk of exposing the business to the year 2000 failure is coming, all the risks must be address in order to reduce the probability of loss. The most effective is to test out. Types of Risks associated with the year 2000 computing crisis are:

- Inaccurate data
- Unauthorized Transactions
- System and Information Vulnerability

- Loss of Data/ Processing History
- Loss of Processing Continuity
- Unacceptable Client Service
- Substandard Processing Level
- Overly Complex Systems
- Complex System Maintenance
- Overly Complex System Upgrades
- Incompatibility with outside systems
- Unacceptable Performance Levels

Each of these risks could render the system ineffective and cause the organization to incur substantial loss.

To test effectively, identification and evaluation of the risks in the computer system is necessary. Most organizations compromise in the testing to save time and money by defining an acceptable level of risk and create a plan that tests for only that level. By defining acceptable risk, testing becomes economical. It means the organization will only test as much as the organization want to spend for.

Under-testing leads to system defects and over-testing means wasting valuable resources. Most of the problems associated with testing occur from one of the following causes:

- Failure to define testing objectives
- Testing at the wrong phase in the lifecycle
- Use ineffective test Technique

As the cost of testing increases, the number of undetected defects decreases, as illustrate in the figure below. From the Cost-effective perspective, the goal is to test until the cost does not exceed the value gained.

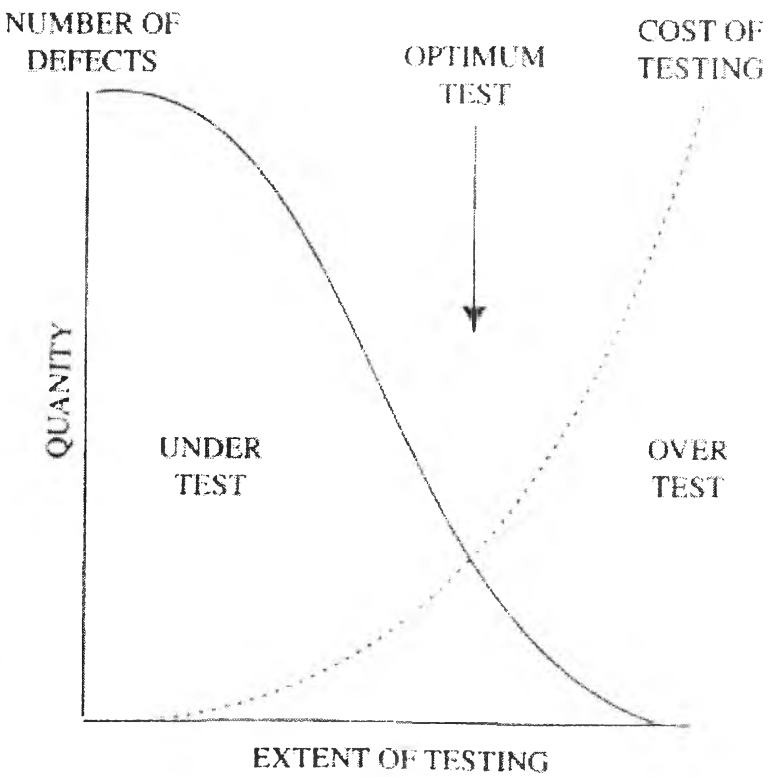


Figure 2.2 : Cost-effectiveness of Testing [9]

Emphasizing cost in the test plan raised 2 concerns: inadequate resources and not enough timeframe. Testing year 2000 Compliance plan required sufficient Time and Resources. Without sufficient resources of testing, the solution might not meet the required risk level. On the other hand, if the project takes too long to implement the solution, the project will increase the risks as the time goes near year 2000.

Year 2000 Risks is triggered by an event. Year 2000 risks associated with the inability of the computer to recognize 2000 as a date. The 5 causes that trigger this risk are:

- Noncompliant internally built software or hardware.
- Noncompliant externally acquired software or hardware.
- Noncompliant embedded software.

- Defective Year 2000 data processing solutions. This is called 'Second Level' defects, this kind of defect will cause problem to the application that interface with it.
- Defective Year 2000 software and hardware enhancements or updates. Normally big organization will take year 2000 opportunity to address known problem the reduce time and resources. However, this can introduce new problem, which result in risk.

The Test strategy should ensure that risk would result only in minimal loss, while optimizing the use of scarce testing resources. With the mismanagement of the time and resources, this could result in another category of risk, administrative, which consist of:

- Poorly skilled system analysts/programmers
- Poorly skilled testers
- Insufficient development time
- Inadequate software and hardware

Software Testing Technique

Verification and Validation

A tester uses Verification methods to ensure the system (Software, Hardware, documentation, and Personnel) complies with an organization's standards and processes, relying on review or non-executable methods. Validation is physically ensures that the system operates according to plan by executing the system functions through the series of tests that can be observed and evaluated. Verification answers the question, "Did we build the right system?" Validation answers the question, "Did we build the system right?"

Verification requires several types of reviews, including requirement reviews, code walk-through, code inspections, design reviews and review reviews. The system user should be involved in these reviews to find the defects before they are building into a system. In the case of purchasing a system, the user input is needed to ensure that

suppliers are aware of the requirement. Validation is accomplished simply by executing a real-life function.

From the Relative cost Versus the Project Phase Diagram, the Verification is in the early stage of the project, which imply to lower cost to the project. The Validation is the tests after system is in place, therefore, the cost will be relatively high. This rule also applied to purchasing cases.

Functional and Structural Testing

Functional Testing do not required the knowledge of the internal logic of the system to develop the test cases. A functional test is the test to simulate the function and observe the results, which is a validation. When conducting functional tests, the tester will use validation techniques almost exclusively.

Conversely, structural testing required knowledge of the internal logic of the system to develop the hypothetical test cases. Structural tests use verification predominantly to confirm the reasonableness of the system by reviewing the structure and logic.

There are advantages and disadvantages for both testing techniques. The matrix below is the summary of the advantages and disadvantages:

	Advantage	Disadvantage
Functional Testing	-Simulate Actual System Usage -Makes no system structure assumptions.	-Potential of missing logical errors in system. -Possibility of redundant testing.
Structural Testing	-Test the system's structure logic -Alternative Testing area beyond Functional test.	-May not met user requirements -Do not simulate real-world situation.

Table 2.1 Advantages and Disadvantages of Functional and Structural Testing

In the real world, testers need to use both methods to test the entire system throughout the project. The tester cannot rely on only one of the method and expect the system to be foolproof. Each of the testing methods is suitable for certain applications. In normal case, structural test would be required before the functional test.

This is the summary of the application or activity that suitable for each testing technique along the project timeline.

Using Verification to conduct Structural Testing:

- Feasibility Reviews. Tests for this structural element would verify the logic flow of a system.
- Requirement reviews. These reviews verify the system relationships of the input and output can handle.

Using Validation to conduct Functional Testing:

- Unit Testing. These tests verify that the system functions properly.
- Integrated Testing. The system run tasks that involve more than one application, database or activity to verify that it performed the tasks accurately
- System Testing. The tests simulate operation of the entire system, and verify that it ran correctly.
- User acceptance. This real-world test means the most to business; and, unfortunately, there is no way to conduct it in isolation. Once the system is interact with external system, that would be the time to verify it functions properly.

The Testers' Workbench

The most common and efficient testing process is the testers' workbench. The workbench is a process of illustrating and documenting how a specific activities is to be performed. It is consists of four components:

- Input. The information need to begin the phase.

- Procedures to do. The physical tasks that will transform input into output.
- Procedures to check. Testing to determine the output meet the standards.
- Output. The exit criteria or deliverables produced from the workbench.

Figure below illustrates the testers' workbench. The solution development cycle, which is composed of many workbenches as a series of tests between pro ect start and when the corrected solution is placed into the production.

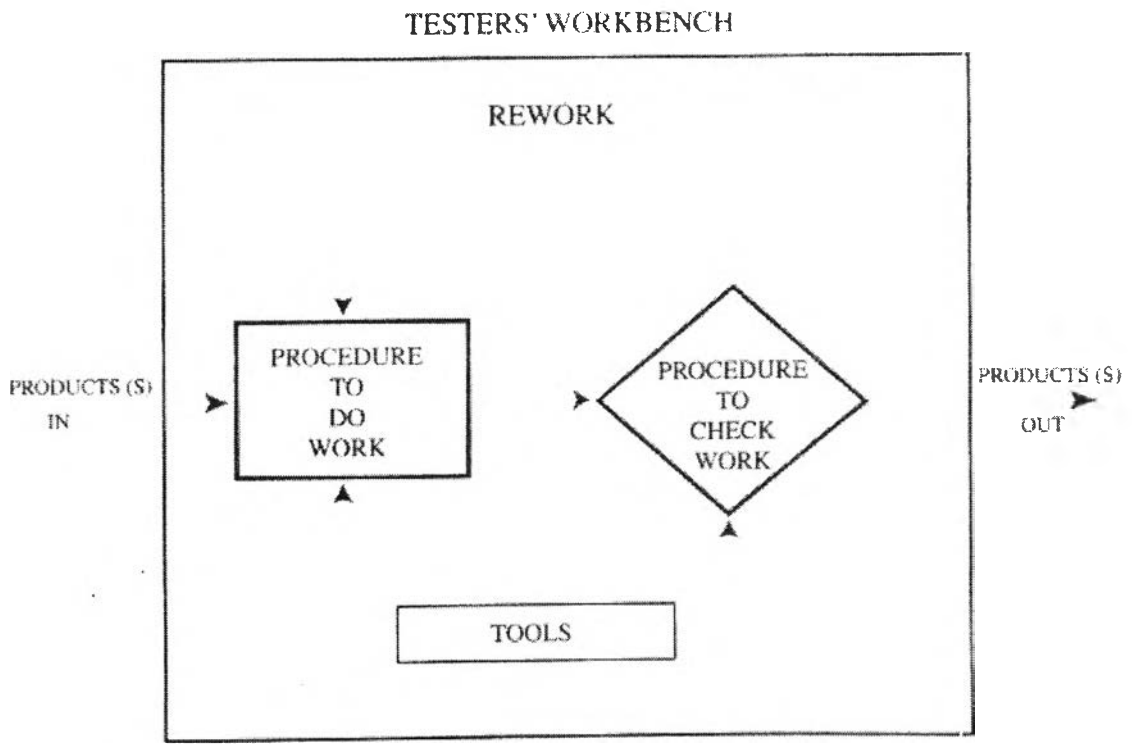


Figure 2.3 : Testers' Workbench [9]

The do and Check procedures are the core of the testers' workbench, and are proportional to each other; the more do procedures a phase has, the more check procedures have to perform.

With the testers' workbench concept, all the work tasks have to go through Do procedure and Check procedure or Correction procedure and Testing Procedure. This is so called V-Testing Concept. The project of correction and testing will slowly converge from start to finish. This indicates that as the correction team attempts to implement a

solution, the Test team concurrently develops a process to minimize or eliminate the risk. The more that these 2 groups are working close together, the high level of risk at the project inception will be reduce to an acceptable level by the project's conclusion.

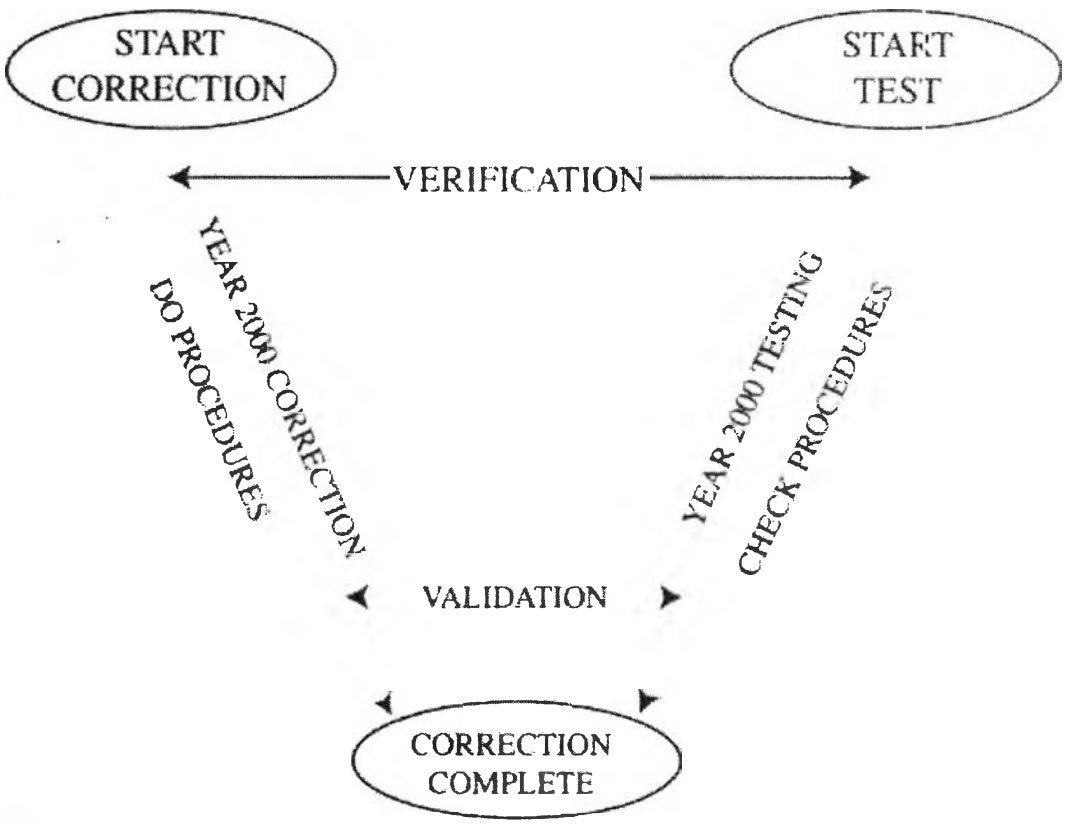


Figure 2.4 : V-Testing Concept and Verification & Validation in Year 2000 testing [9]

9 Steps of Year 2000 Compliance Testing

The year 2000 Testing Process follows the V-Concept of Testing that mentioned beforehand. The first 4 steps use verification in conjunction with structural testing and the last 5 steps use validation in conjunction with functional testing.

1. Verify Year 2000 Assessment. Assess the scope of the year 2000 computing crisis. This is a prerequisite to determining the efforts required to correct the problem. During this step, testers will challenge the completeness and correctness of the assessment performed by the project team. At the end, both

team should know how much time and resources they'll need to test and correct the Year 2000 Problem.

2. Develop the Year 2000 Test Plan. Forming the test plan for year 2000 Testing will follow the same pattern as any software planning process. The structure of the plan will also be the same, but the content will vary depending on the amount of the hard coded chips and vendor software of company uses.

3. Verify Suppliers Compliance Compatibility. The test team must determined whether vendors will update the software or hardware they provide to the team on their own. However, the team still need to check whether the updates are theoretically compatible with the systems.

4. Verify Internal Compliance Compatibility. The year 2000 Compliance team need to develop a plan that defines if and how internal systems can comply with the year 2000. The more detailed the plan, the easier it will be for the test team to verify the adequacy and completeness of the plan through a review of process that logically tests every element of the plan.

5. Inspect implementation deliverables. This step will walk the test team through the inspection of the corrected software and hardware prior to execution. At this step, the team need to physically validate every application and system to identify the defects. This will be time consuming, but will keep cost down; removing the defects through unit inspection or system testing later in the project is more expensive.

6. Perform system testing of changes. This step identify the test data necessary to perform the effective year 2000 testing. It describes how to break down the test plan into specific test activities that needed to validate the performance of the operational system.

7. Perform acceptance testing of the changes. System users and customers need to validate the system by using it in a real world setting.

8. Prepare for Year 2000 Reports. The test team must create interim and final test reports both for fellows testers and the management. Testers will need to know the testing and defects identification and correction. In order to provide the management with updates on the status of the overall project, along with the assessment of risks the organization faces.

9. Monitor Year 2000 implementation Changes. This step shows testers how to maintain and control their projects in the face of the extensive system upgrade, changes in the data input processes, version control, and testing process control in a dynamic change environment.

Year 2000 Compliance Phases (Correction)

The Year 2000 Life Cycle is a process for addressing Year 2000 issues. The following is an example of a model composed of five phases, each representing a major Year 2000 activity. Both the private and public sectors have used this model in addressing their respective Year 2000 issues. The five phases are described simply and very briefly as follows:

- *Planning and awareness:* Define the Year 2000 problem and gain executive level support and sponsorship for establishing the problem as a high priority item for resolution. Research and establish a project plan, and obtain budget and resources. Note that the planning activities are also relevant to the other phases described below.

- *Assessment (inventory):* Evaluate the Year 2000 impact on the enterprise; develop contingency plans to handle data exchange issues and system failures (dysfunction or system crashes); prioritize systems by identifying those that are critical.

- *Remediation (renovation):* Convert, replace, eliminate, or work around one or more system elements; Modify interfaces.

-*Validation:* Test, certify, and validate all system elements that have been converted or replaced.

- *Implementation:* Place into production all system elements that have been converted or replaced.

This Recommended Practice does not address the subject of validation within the Implementation Phase, which is generally outside the scope of testing/validation organizations. However, due to potential interdependencies between any remediated/validated system and the operational environment within which it must function, there is a likelihood that additional problems will emerge in the Implementation Phase. The origins of these problems could be in the remediated/validated system, in some other system(s) in the environment, or in the interactions among them. Therefore, it is recommended that an effort be made to plan, audit and manage the Implementation Phase with the goal of detecting, locating and addressing such problems.

Remediation Technique

This concept embodies one or more processes to repair or eliminate malfunctions relating to Year 2000 date data processing under a predetermined set of criteria. The following is a non-exhaustive list of techniques or strategies that have been employed in the remediation process. These techniques are not necessarily mutually exclusive and not necessarily sufficient in and of themselves to solve the problem:

- Bridge - employing a date data bridge that converts date formats
- Elimination - retiring the system
- Encoding - encoding four-digit year information into an existing field
- Field expansion - expanding the year field to four digits
- Replacement - replacing one or more system elements
- Windowing - using a 100-year logic window

There is general agreement within the industry that there is *no single* method of remediation that can be applied to all situations. In any given situation, the method chosen will depend on the operating environment and other factors such as the prevalence of interfaces with other applications and the amount of date data processed within or between applications. The following methods include some of the most

common fixes being implemented in the industry. This list does not describe a number of low usage or proprietary remediation methods in use.

Bridge technique

This is a conversion mechanism that changes data elements to reconcile format differences between system elements. Date-related bridges format dates in such a way that they can be accepted and acted upon properly. For example, a bridge might be a program that is invoked to change a two-digit date field in a sending application to a four-digit date field in a receiving application when date data is transferred between the two system elements.

Elimination technique

This is the retirement of a system or application no longer deemed necessary.

Encoding technique (also called encryption, offset counter format, or integer date format)

Unlike field expansion, encoding allows current field sizes to be maintained by storing additional information into existing fields. A more efficient use of bits may allow inclusion of century information. This may be accomplished, for example, by using unused bits in an existing representation to encode century information, or by converting the data type from ASCII to binary in order to allow larger numbers to be represented in the same field. Similarly, if the numbering system was changed from decimal to hexadecimal, two-digit year values greater than 99 could be stored.

Field expansion technique

This is a technique that converts existing data and programs by lengthening the year fields from two-digits to four-digits.

Replacement technique

This is the retirement of a system or system element that is not Year 2000 compliant and replacement of it by another system or system element that is Year 2000 compliant.

Windowing technique (also called logic fix)

This is any of several procedural techniques based upon the addition of logic that uses a specified 100-year interval to interpret a two-digit year value as an unambiguous four-digit year. The procedural techniques commonly used are typically categorized into 'Fixed Windowing,' 'Movable Windowing,' and 'Sliding Windowing.' Use of this method of remediation depends on knowledge of the date format used across interfaces between two or more system elements;

- The pivot year used
- The method of changing or 'sliding' the pivot year, where applicable; and
- A description of the error handling/reporting when exchanged date data is not in the expected format.

Fixed windowing

This is a procedural technique in which two-digit year values are interpreted within a fixed 100-year window. The window is typically documented in terms of a range of years (for example, 1950 to 2049) or in terms of a pivot year (for example, a pivot year of 1950 causes values between 50 and 99 to be interpreted as 1950 to 1999, and values between 00 and 49 to be interpreted as 2000 to 2049).

Movable windowing

This is a procedural technique in which two-digit year values are interpreted within a 100-year window, which may be user- or installation-specified. The range of years in the window or the pivot year can be specified when the system is installed or started.

Sliding windowing

This is a procedural technique in which two-digit year values are interpreted within a 100-year window that is defined in terms of the current date. With this technique, as the current date moves forward, year to year, the window also moves or 'slides' forward. Sliding windows are sometimes documented by specifying a value (usually called a 'slider') that, when added to the current year, defines the pivot year of the 100-year window. Thus, for example, a sliding window operating on a current date of 2 February 1998 (1998-02-02) with a 'slider' defined as '-40' would result in a pivot year of 1958 (1998-40) and a window range of 1958-2057. On 2 February 1999 (1999-02-02), the pivot year would be 1959 (1999-40) and a new window range of 1959-2058.

V-Concept with Year 2000 Correction and Testing Process Flow

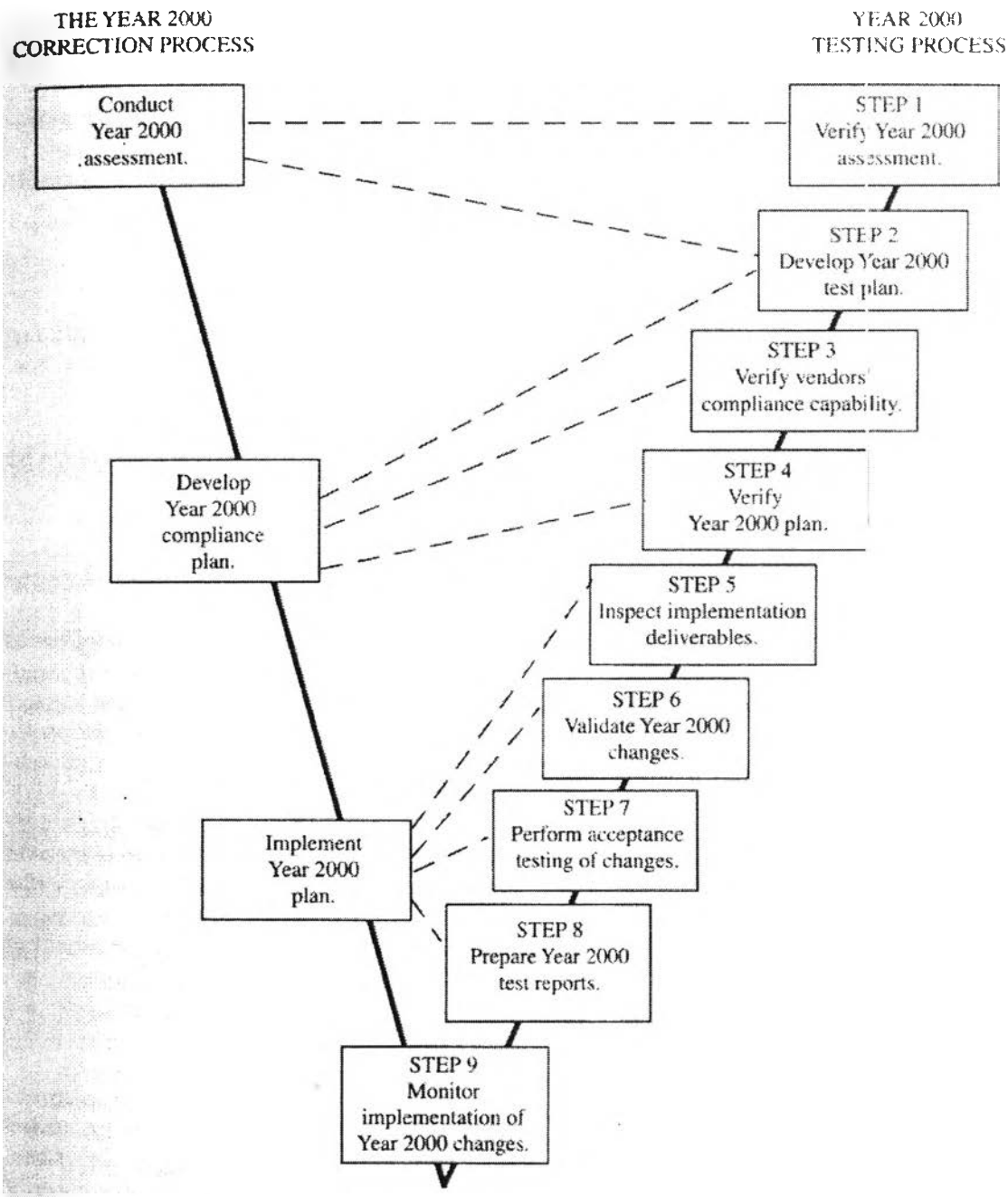


Figure 2.5 : V-Concept with Year 2000 Correction and Testing Process Flow [9]

Year 2000 Support Tools

In addition to the in-house testing techniques, outside vendors can supply tools specifically designed to support the test team, implement and test the solution. Big organizations that use many date-sensitive applications may need to make a lot of changes and it would be nearly impossible for the team to implement them without the tools or otherwise delayed the project. Organizations should work with tool vendors to identify the appropriate tool for their projects. By doing so, organizations can fully utilize the limited resources to drive the project and delegate the tasks outside their specialization to external vendors. The list below is the sample of the Year 2000 support tools, listed by type.

Tool Type	*Tool Capability*
Data Generation	Preparation of Test Data for year 2000; Expected to be a significant effort via the automated generation of test data.
Test Case Management	Test Case management can reduce effort to develop current century, cross-century, and next century test cases.
Record/Playback Capability	Recording inputs for future playback reduces retesting efforts.
Data Aging	Preparation of test data for Y2K testing, via the ability to advance dates in test data transactions.
Clock Simulators	Ability to simulate dates other than current system date.
Test Coverage Analysis	Quantifies program logic execution during test.
File Comparisons	Component outputs from files produced from a system before and after year 2000 corrections.
Date Search	Locates date routines.

Table 2.2 : Sample List of Year 2000 Support Tools Testing [9]

Date Calculation & Sematech Year 2000 Test Scenarios

The system or criteria that contains Date are in Hardware, Software and Relationships. The list below is the guideline that can be use to construct the checklist of the Hardware, Software and Relationships (Interfaces) Inventory for Year 2000 Tests.

List of Hardware

Computer Systems:	Mainframes, Minis, Servers, Workstations, PCs, Portable Devices
Peripherals:	Storage Devices, Printers, Scanners, Display Units, Consoles, Telecommunication, Fax, Modems
Network Systems:	Routers, Firewalls, Repeaters, Bridges, Gateways, Switches, Hubs, Network Services such as network time, DNS and others Embedded Systems
GPS :	Receivers, Ground Support Units, Navigational Systems, International Funds Transfer Synchronization
Facilities:	Electrical Supply, Measurement, Control and Protection Systems, Access Control Systems, Card Readers, Air Conditioning, Lifts, Alarms, Fire Detection & Suppression Systems, Security / Environmental Monitoring Systems, Lighting, CCTV

POS & Retail Systems : POS Scanners and Cash Register Units, Credit Card Systems

Manufacturing :	Plants, Production Lines, Automated Machinery, CNC Process Controls, Robotics, HVAC Systems, Simulators and Automated Storage & Retrieval, Compressor Control Systems, Panel Mounted Devices, Field Devices, Lab / Analytical Equipment, MultiLoop Telemetry, SCADA and DCS
Transport :	Vehicles, Fuel Services, Traffic Control, Radar Systems, Ticketing Machinery
Communications :	PABX, Digital Phones, Mobile Phones, Fax, Telephone Switches, Satellites, Signal Broadcast Stations, Cable Systems
Medical Systems :	Heart Defibrillators, X-ray Equipment, Pharmaceutical Dispensing Systems, Infusion Pumps, Kidney Dialysis Machinery, Life Monitoring & Support Systems
Office Equipment :	Copiers, Time recording Systems, Fax, Postage Franking, various AV equipment, Pre Printed Forms, Timed Lock Safes

List of Software

Operating System	NOS, Password Security, Access Control, File System, Queues, Scheduler, Backup & Restore
Middleware	Development Tools, Compilers, Libraries, Database Engines, Transaction Monitors, Management Tools & Platforms
Applications	Date Sensitive Interval Logic (EOD, EOW, EOM, EOY)
	Day of Week Computation
	Age Calculation
	Interest Calculation
	Forecasting / Planning (incl spreadsheets, OLAP, ERP, DSS)
	Data Manipulation wrt Date Stored
	(e.g. sort or select by date, expiry, aging, validation)
	Date Representation for Display or Print Output
	Permanent Storage of Dates
	UIN / FIN Prefix for NRIC and other date encoded serial numbers
	Utilities
	Messaging / Mailing / Workflow systems

Relationships Among Assets with Y2K Dependencies :

Interfaces	Parameters, Data Files, Control Files, Data Structures, Variables, IPC, RPC, Drivers, Configuration / Initialisation Files / Registries, Archived Data
Interactions	EDI, e-Commerce, Internet-Commerce, File Transfers, Embedded Dates in Serial Number from Bar Code Scanning, other Contracted Service Providers

Note : Only ANSI 4010 and EDIFACT 97A EDI standards are Y2K ready.

From the list above, they can be categorized into categories. There are four areas that containing the date and can possibly affect the date interpretation and calculation directly or indirectly. These four areas are:

1. Storage

2. Display
3. Input
4. Calculation

Within these areas, it is necessary to identify the answer for the questions below.

Storage:

How are dates stored by the app (data structure names, types, etc.)

What are the logical limits?

Are there conversion functions to go back and forth to other date storage (e.g., system date, server, ...).

What are they?

Display:

How are the dates formatted for display?

Are system calls used for formatting dates? Which?

If your application "rolls its own" code for display, does it reference any system setting to change display. Where/how can it be changed?

Input:

Where are dates accepted as input?

Are they stored as dates or strings or...?

What are the date parsing rules?

What are the dependencies for the date parsing functionality (System date? OLEAUT32, reg settings?)

More specifically, what are the rules for converting both to and from two digit years? What are the dependencies for evaluating those rules?

Do users or the system ever enter a special date that has an understood meaning different from an actual date, such as 9/9/99?

Calculations:

What date calculations are performed?

Are their type conversions required in order to perform the date calculations?

Are they consistent with the rules for parsing date input?

Is there a leap year calculation function?

Is it correct (usually small enough to be easily walked through)?

Where does it get used? Note for tests later: if serial dates are used, calculations such as adding, subtracting and comparing dates are less likely to be problematic, but more leap year type problems can occur. If dates are stored with separate Month, Day and Year fields, the opposite is true.

What logic is there for handling date calculations at the limits?

Date calculation in this case is not just the year 2000 rollover date, but also involve is many possible misinterpret date in our calendar for next several years. Some companies or Organizations use Year 2000 scenario to address the entire date interpretation problem at one time. With this action, they would be able to ensure there is no failure for the operation in near future.

In many Year 2000 articles are always discussing about the Julian and Gregorian Date. Gregorian Calendar is the modern calendar that accept internationally (Date and Time Format is DD/MM/YYYY). For Julian calendar, there are many definitions of Julian days and Julian dates. The formal definition of a Julian Date is not used in this research because of the complex historical adoption of the Gregorian (modern) calendar. Instead, an informal Julian day format is used and is defined as the ordinal day of a year, which is still using in some organization.

Most of American Government agencies and contractors commonly use the informal Julian day. The informal Julian day format that widely used is the ordinal day of a year (for example, Julian day 032 represents February 1st, or the 32nd day of the year). Variations on informal Julian day formats include using a preceding two-digit year (for example 96032 for 2/1/96) and separating the year with a dash (for example 96-032). Another, less popular, Julian day format uses a one-digit year (for example 6-032). These additional formats do not uniquely identify the century or decade. There are

consequences when using these formats; for example, the Julian day 00061 can be interpreted as 3/1/2000 or 3/2/1900.

This table below is Date and Reason that possibly make the computers systems and other technologies to misinterpret the time format.

Event Horizons (YYYY-MM-DD)	**Cause & Potential Impacts**
1900-01-01 (Monday)	The number of days in a century is not evenly divisible by 7
1999-01-01 (Friday)	End Of File magic number terminates SAP R/2 4.4
1999-09-01 (Wednesday)	999 on the Julian calendar. In many computer programs 999 denotes "out of range."
1999-04-09	9999 on the Julian calendar. In many computer programs 9999 denotes "end of input."
1999-04-10	Julian date 9999 Magic Number for End-Of-File (EOF) sentinel
1999-08-22	GPS EOW; most navigation systems may fail
1999-09-09 (Thursday)	9999 on the Julian calendar. In many computer programs 9999 denotes "end of input."
1999-09-10 (Friday)	In systems that have used 9-9-99 as a never expire date, logic allowing deletion of data after a specified date.
1999-10-01 (Friday)	This is the first day of the US Government Fiscal Year FY2000.
1999-12-30	Last business day in 1999 for many markets.
1999-12-31 (Friday)	Last day in 1999 year.
2000-01-01 (Saturday)	The first day of the Year 2000.
2000-01-03 (Monday)	This may be the first business day of the Year 2000.
2000-01-04 (Tuesday)	This may be the first business day and first banking day in the Year 2000.
2000-01-07 (Friday)	This is the first Friday in the Year 2000.
2000-01-10 (Monday)	This is the first 7-digit date after Year 2000 rollover.
2000-01-17 (Monday - Martin Luther King day - USA holiday)	This may be the first Monday holiday in the Year 2000.
2000-01-31 (Monday)	This is the first month-end day in the Year 2000.

2000-02-28 (Monday)	This date is not expected to cause any specific Year 2000 errors. Its relevance to testing is that it should be used as a start date in testing the system ability to increment to 2000-02-29.
2000-02-29 (Tuesday)	The Year 2000 is a leap year.
2000-02-30 (Non-existent)	This day does not exist. Date functions should continue to recognize this as an invalid date.
2000-03-01 (Wednesday)	This is the first day after leap year day.
2000-03-31 (Friday)	This is the last day of the last month in the first quarter of the Year 2000.
2000-04-03 (Monday)	This is the first business day of the second quarter of the first year in the Year 2000.
2000-04-17 (Monday)	Primary U.S. Income Tax due date in the Year 2000.
2000-04-28 (Friday)	April is the first business month whose last day coincides with a weekend in the Year 2000. This is the last business day of April.
2000-04-30 (Sunday)	This is the first month-end that coincides with a weekend in the Year 2000.
2000-06-30 (Friday)	This is the last day of the last month of the second quarter.
2000-09-29 (Friday)	Last business day of the third quarter in the Year 2000.
2000-09-30 (Saturday)	This is the last day of the government fiscal year and last day of the third quarter of the Year 2000.
2000-10-01 (Sunday)	This is the first 7-digit date with a 2-digit month value.
2000-10-10 (Tuesday)	This is the first date, after rollover that must be represented as an 8-digit date.
2000-12-31 (Sunday)	The last day of the Second Millennium of the Gregorian calendar.
2001-01-01 (Monday)	This is the first day of the third Millennium on the Gregorian Calendar.
2001-01-05 (Friday)	This is the first Friday in the year 2001.
2004-02-29 (Sunday)	First leap day after Year 2000 rollover not affected by a century or millennium transition.
2004-12-31 (Friday)	This date can be used to determine if normal leap years are recognized by an ordinal date system.

2011-01-01	some Microsoft application products will fail due to the method used to resolve YEAR 2000 issues. (i.e., year > 10 assumed to be in 20th century)
2020-01-01	Microsoft Excel 95 when YY is used
2038-01-19	32-bit Unix upper limit of signed integer for date

Table 2.3 : Potential Date Calculation Problem

However, there are so many dates that can possibly cause malfunction in some systems but does not in the others or it is not in the time interval of interest. With this fact, the Semiconductor Manufacturing Companies aware and come up with the Set of Tests that will address the problem that is general enough for all the Semiconductor Manufacturing Industry Equipment to use as a guideline for testing their year 2000 compliance.

Sematech (See Appendix A for detail of the organization) Year 2000 Test Scenarios is the tests for year 2000 compatibility in Semiconductor Manufacturing Environment. All the tests are basing upon the common dates that possibly cause the computer with date calculation to misinterpret, behave and perform in different manner that was customized for Semiconductor Industry. In this test scenarios is testing these primary potential misinterpret dates that agreed among Sematech Members:

- December 31, 1999 to January 1, 2000 (century change)
- February 28, 2000 to February 29, 2000 (leap day)
- February 29, 2000 to March 1, 2000 (leap day + 1)
- December 31, 1998 to January 1, 1999
- December 31, 2000 to January 1, 2001
- September 8, 1999 to September 9, 1999
- October 10, 2000

Sematech Year 2000 Test Scenarios are base on the dates mentioned above and the four areas that have date related, namely, Storage, Display, Input and Date Calculation. All of the test scenarios will have to do test on the system on these categories:

- Basic rollover and Hold the Date after Shutdown - Some systems having the problem that they cannot rollover properly or even they can rollover but they cannot keep the correct date after restart. This test has main concern over the system time memory, in most case CMOS and BIOS, to see if system fails to rollover and keep the date after event horizon.

- Date Set and Hold using the Correct Calendar – This test purpose is to identify the capability of the system to set the date after the event horizon and hold that date with correct calendar. This test will not take the previous test into consideration. This is because the previous test is a separate issue. This test is not using the rollover but to set it at the date after event horizon to test the performance, functionality, and integration of the platforms, applications, databases, utilities, and interfaces in an operational environment.

- Ability to execute the application that straddles the change from 1999 to 2000 – This test is use to demonstrate the system capability to operate with normal manner during the transition period. Some system having different time format for the time before year 2000 and after or other event horizon. With this operation, the system might have problem due to different time format in the same operation.

- Ability to Provide Equivalent feedback based on activities before and after 2000 – The system needs to demonstrate the accuracy of the feedback of the same process or operation requested for both before and after year 2000. This test is the extension of the concern about the time format that it may make the feedback from system behaves in different manner.

- Purge/Retention Routine – This test is use to ensure the Purge or Retention process of system to operate according to correct calendar to avoid data lost for the required date or flushed storage area with unnecessary data.

- Request time format – This test is mainly use for the Network Capability to ensure the proper communication between systems with the mutual understandable time format.