목전의 "서기 2000년" 혼란과 대책

2000년 1월 1일이 되면, 어쩌면 증권, 은행, 보험 등 모든 금융 시스템은 마비되고 심지어는 엘리베이터나 세탁기까지 엉뚱하게 작동하는 대 혼란이 발생할지도 모른다. 이러한 심각한 우려는 점술가나 사이비 종교지도자의 혹세 무민적인 예언이 아니라, 미리미리 대비하지 않을 경우 현실로 닥칠 필연적 사실이라는데 더 큰 문제가 있다.

신.종 철 (주)송우정보 대표이사 · 전북산업대학교 교수

2000년, 무엇이 문제인가

1946년 ENIAC 컴퓨터의 개발로부터 시작하여 1900년대 중반부터 널리 활용되기 시작한 각종 전산 시스템은, 초창기 메모리와 디스크의 사용량을 절약하기 위하여 연도 표기방법으로 1996년을 96년으로 표기하는 두자리 표시방법이 채택되어 지금까지 일반적으로 사용되고 있다. 그러나 서기 2000년이 되면 이러한 표기 방법이 더 이상 사용 불가능해 지기 때문에 여러 가지 문제점이 발생하게 된다.

- ○ 모든 조회화면이나 보고서 상에서 2000년은 00으로 표시된다.
- ○ 00은 99보다 적기 때문에
  - – 대부분의 정렬(Sort) 프로그램이 제대로 작동되지 않을 것이며
  - – 두자리 년도를 사용하는 모든 비교나 연산조작이 엉뚱한 결과를 산출할 수 있다.
- ○ 많은 프로그램들이 99보다 큰 년도는 없다고 가정하거나, 서기 2000년을 윤년으로 처리하지 못하거나, 혹은 00을 잘못된 입력으로 간주하여 에러로 처리할 수 있다.
- ○ 20세기를 미리 가정하여 아예 '19'가 프로그램에 내장되어 사용되고 있다.

조사회사인 가트너 그룹의 분석에 따르면, 이 문제를 해결하는 데에는 프로그램 한 라인당 평균 $0.5~$1.0이상의 비용이 소요되어 서기 2000년까지 전세계적으로는 최고 6천억달러(약 480조원)정도의 어마어마한 비용이 필요할 것으로 예상되고 있다.

더구나, 이러한 문제점들은 COBOL 이나 파일 구조에 국한된 문제가 아니라 심지어는 시스템 클럭까지 연관이 있을 수 있고, 서기 2000년 1월 1일 00:00 이전까지는 반드시 해결되어야 할 뿐만 아니라, 불행하게도 세금이나 죽음처럼 아무도 피해 갈 수 없다는 사실이 문제의 심각성을 더해주고 있다.

해결방안은 무엇인가

근본적으로 이 문제는 새로운 프로그램의 개발이 아니라 기존 프로그램의 유지보수에 관련한 업무라는 사실에 유의할 필요가 있다. 다시 말하면 이 문제를 해결하기 위해서는 유지보수 측면에서 접근해야 하며, 이것은 바로 자기 스스로가 아니면 아무도 완벽한 해결방안을 제시할 수 없다는 사실을 의미하기도 한다. 그러나 수십 · 수백만개의 프로그램을 보유하고

있는 회사나 기관이라면 어디에서부터 손을 대고, 어떻게 해결해 나갈 수 있을 것인지 막막한 문제가 아닐 수 없다.

이 문제에 대한 일반적인 해결방안은 :

(1) 새로 개발하는 시스템은 연도를 4가지로 표시하도록 한다.

현재 개발하고 있는 시스템이나 새로 개발한 시스템은 설계단계에서부터 프로그램이나 데이타 파일에서의 연도 표시를 모두 4자리로 (YYYY/MM/DD) 표시하여, 서기 9999년까지는 이러한 문제가 아예 발생하지 않도록 개발한다.

(2) 전문가의 도움을 받아 파일로트 프로젝트를 먼저 수행한다.

방법론을 습득하거나 유용한 틀들을 선정하고 이를 익히기 위해서는 외부 전문가의 도움을 받아 Pilot시스템을 수행해 보는 것이 가장 좋은 방법이 될 수 있다. 어차피 대부분의 프로그램을 자체적으로 수정하고 테스트해야 하기 때문에, 초기에 Pilot 프로젝트를 통하여 관련 기술을 습득하고 내부인력을 양성하는 것이 시간과 비용을 절약할 수 있는 지름길이 될 것이다.

(3) 유용한 틀들을 활용한다.

근본적으로 이 문제는 기술적으로 어려운 문제는 아니다. 하지만 대상 프로그램이 방대하고 시스템이 복잡하기 때문에 유용한 틀들을 선정하여 활용하는 것이 무엇보다 중요하다. 가트너 그룹의 분석에 의하면, 적당한 틀과 방법론을 사용하면 수작업에 비하여 절반정도의 노력과 비용으로 이 문제를 해결할 수 있을 것으로 기대되고 있다.

(4) 가급적 빨리 시작한다.

사실 이 문제는 서기 2000년이 되기 전에, 빠르면 내년 하반기부터라도 발생하기 시작할 가능성이 있다. 특히 장단기 예측이나 신탁, 보험등을 다루는 응용프로그램들은 당장이라도 문제를 일으킬지 모른다. 뿐만 아니라, 이 문제는 한계시점이 명확하게 정해져 있어 회피할 수 없는 문제이기 때문에, 주어진 인력과 자원으로 이 문제를 해결하려면 서기 2000년에 가까워질수록 소요되는 비용과 노력은 점점 증가할 수밖에 없다. 따라서 2000년이 오기전에 전산직을 떠날 무책임한 당신이 아니라면 하루라도 빨리 계획을 수립하고 착수하는 것이 바람직할 것이다.

클라이언트/서버 시스템은 안전한가

일반적으로 클라이언트/서버 시스템은 업무 단위로 분산되어 있을 뿐만 아니라, 대부분 최근에(보통 2~3년 이내) 개발되어, 상대적으로 문제가 덜 심각할 것으로 예상된다. 그러나 하나의 클라이언트에서 잘못이 발생할 경우 그 영향이 다른 시스템 까지 연쇄적으로 미칠 수 있고, 메인프레임과는 달리 시스템 전반에 걸친 분석 및 평가를 위한 유용한 수단이 별로 없으므로, 발생 가능한 문제점들을 주의깊게 살펴보고 확인하여야 한다.

- ○ 서버의 DB에서는 연도를 4 digit로 표식되어 있으나, 클라이언트 프로그램에서는 2digit로 사용하지는 않는가?
- ○ 구형 PC인 경우, BIOS가 4 digit 연도 표시를 지원하는가?
- ○ 메인프레임과 연결된 3-Tier 클라이언트/서버 시스템인 경우, 호스트용 프로그램이나 파일이 수정되면 그 영향은 제대로 반영되는가?
- ○ 서기 2000년을 윤년으로 처리하지 않는 프로그램은 없는가?
- ○ 수정된 프로그램의 배포 및 관리는 적절하게 이루어지고 있는가?

실무자를 위한 몇 가지 힌트

이 문제에 대한 접근 방법을 찾아 고심하고 있는

프로그래머나 전산관리자를 위하여 지금까지 알려져 있는 몇 가지 해결방안 및 툴들을 소개한다.

(1) 접근단계의 구분

제 1 단계 : 문제의 이해 – 계획 및 영향 평가 (Impact Analysis)

– 보유 프로그램 조사 /분류 및 범위 결정

– 사용기술 및 프로그램 대체 전략을 포함하는 변환방법 결정

– 프로젝트 실행계획 수립

제 2 단계 : 해결방안 기술 – 확인(Assessment) 및 예측(Estimation)

– 프로그램의 구조 및 시스템 구성 분석

– 소요 작업량 예측 및 자원소요 /제반사항 결정

제 3 단계 : 문제의 해결 – 실행(Implementation) 및 검증(Verification)

– 프로그램 및 데이타 수정

– 수정된 시스템의 테스트 및 검증

(2) 대상 프로그램 확인

- 자체개발한 프로그램 – 사용언어, 데이타 화일 등
- 외부 구매한 프로그램 – 공급자가 아직도 계속 지원을 제공하는가, 소스 코드는 보유하고 있는가 등
- PC용 BIOS – 예를 들면 인텔팬티엄 BIOS 버전 12는 2000년 지원이 어려운 것으로 알려지고 있음
- 마이크로 프로세서 시스템 – 엘리베이터, 구내전화 시스템 등

(3) 예상 소요비용 산정

○ 산출기준

– 가트너 그룹 : 라인당 $0.5~$1.0 이상(프로그램당 $450~$600)

– 앤더슨 컨설팅 : 100만 라인당 2~4Man /Year 이상

○ 단계별 소요비용

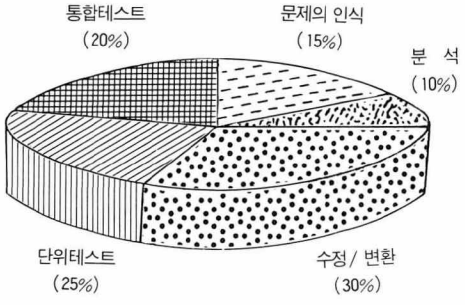
– 문제의 인식 : 15%

– 분 석 : 10%

– 수 정 : 30%

– 단위 테스트 : 25%

– 통합 테스트 : 20%



출처 : 마이크로 포커스

(4) 수정(Conversion)방법

○ 2digit → 4digit 변환

연도 표시를 두자리에서 네자리로 변경하는 방법으로, 2000년도 문제를 해결하는 가장 간편한 방법중의 하나이다. 이 방법은 연도표시 길이만 변경하므로 프로그램의 로직을 변경할 필요가 없어 테스트 부담이 적고, 기존 시스템과 병행하여 테스트할 수 있는 장점이 있다. 그러나 관련되는 모든 데이타 파일을 수정해야 하고 사용하는 디스크 용량이 늘어나며, 경우에 따라서는 블록 크기나 비퍼 크기를 조정해야 하는 등 적용하기 어려운 경우도 있다.

○ 날짜 구분점 설정 (Date Windowing)

100년 단위의 날짜 구분점을 설정하여 프로그램 내에서 적용하는 방법으로, 예를들어 1950년부터 2049년까지를 구분점으로 설정

하는 경우, 프로그램 내에서 50보다 큰 숫자는 1900년대로, 그리고 50보다 적은 숫자는 2000년대로 간주하여 처리하는 방법이다. 이 방법은 연도 표시 자릿수의 확장을 피할 수 있으나, 데이타에 대한 외부 시스템에서의 공동 활용이 어렵고 회사나 기관이 전체적으로 통일된 기준을 적용하여야만 성공할 수 있다. 데이타 파일을 수정하지 않고 그대로 사용할 수 있으며 디스크용량이 추가로 늘어나지 않는 장점이 있으나. 프로그램 로직을 수정해야 하고 이에 따른 테스트 부담이 늘어나는 단점이 있다.

또 다른 하나의 방법으로, 데이타 파일은 수정하지 않고 단지 가까운 미래를 염두에 두고 연도(YY)에서 일정한 숫자(예를 들면, 30 등)를 뺀다음 그것을 비교나 연산 등에 사용하는 방법도 있다.

○ 날짜 엔코딩 (Encoding)

이 방법은 세기를 표시하는 바이트의 16진수 값을 기존의 데이타 필드에 더하거나 사용되지 않는 1 bit를 활용하여 1900년대와 2000년대를 구분하는 방법이다. 이 방법을 사용하면 데이타 파일의 자릿수를 변환하거나 추가적인 디스크 용량을 사용하지 않고서도 프로그래밍 문제를 해결할 수 있는 장점이 있으나, COBOL의 Packed Decimal 필드(COPM-3)에서는 작용할 수 없고, 응용프로그램내의 처리 절차가 복잡하여 특별한 테스트가 필요하게 되는 단점이 있다.

이외에도, YY/MM/DD 값을 서기 1990년 1월 1일을 기준으로 한 누적 경과일로 계산하여 파일에 보관하고, 입출력시에는 이 누적값을 다시 CCYY/MM/DD로 환산하여 사용하는 방법도 있다.

(5) 툴 선정시 유의사항

분석 및 변환작업이 이루어지는 위치에 따라 메인프레임용 툴과 PC용 툴로 구분할 수 있는데, 일반적으로 호스트 컴퓨터의 부하 증가를 방지하고, 응용 프로그램의 Down-Time을 줄이며, 나아가 GUI 등 다양한 기능들을 활용하기 위해서는 PC로 다운로드하여 처리하는 방법이 보다 유리하다고 할 수 있다.

툴은 프로그래머의 생산성 향상은 물론, 나아가 변환작업 전체의 성패까지 좌우할 수도 있다. 따라서 선정시에는 다각적인 검토가 필요한데, 몇가지 유의사항을 살펴보면 다음과 같다.

○ 툴의 종류
○ 가격 및 기능 범위
○ 사용의 편이성 및 친숙도
○ 추후 활용성
○ 시스템 부하 증가량
○ PL/1의 지원 여부

(6) 검토 및 실행 단계의 착안사항

- 수정작업을 수행할 인력자원은 충분한가?
- CPU의 부하량은 추가적인 수정작업을 감당할 수 있을 만큼 충분한가?
- 대용량 파일이나 데이타베이스 처리를 위하여 디스크 용량은 충분한가?
- 응용프로그램의 많은 부분이 변경될 경우 테스트 작업은 어떻게 수행한 것인가?
- SAS, FOCUS와 같은 외부 툴로 부터의 입력은 어떻게 처리하고 테스트할 것인가?
- PC에 있는 사용자의 데이타 및 프로그램 수정은 누가 지원할 것인가?
- 버퍼 크기와 맞지 않는 확장된 레코드로 인하여 발생할 수 있는 실행속도 문제는 어떻게 대처할 것인가?

(7) 툴 및 해결방안 공급회사

○ 변환서비스 제공 – Alydaar Software Corp.
Peritus Software

Services Inc.
Cogni-CASE

○ Date Routines - TransCentury Data System

○ 분석 툴 - Revolve(Micro Focus)
COBOL Analyst (SEEC Inc.)
System Vision Year 2000 (Adpac Corp)
ESW(Viasoft)
Discover(Software Emancipation Technology Inc.)

○ 프로그램 변환 툴 - COBOL Workbench (Micro Focus)

○ 데이타 변환 툴 - File-Aid(Compuware Corp.)
Century Conversion Software(Quintic Systems Inc.)

○ 테스트 툴 - VIA/SmartTest&Validate(Viasoft)
Tictoc(Isogon)
Xpediter & Xchange(Compuware Corp.)

○ 컨설팅 제공 - Micro Focus
EDS
Andersen Consulting

(8) 피해야 할 10가지 Pitfalls

(1) 지금 즉시 착수하지 않고 망설인다.
(2) 변환작업에 착수하기 전에 사전 분석작업을 철저하게 하지 않는다.
(3) 프로젝트 관리 및 인력자원 할당의 필요성을 낮게 평가한다.
(4) 테스트 작업의 범위와 소요비용을 충분하게 책정하지 않는다.
(5) 변환할 프로그램의 우선순위를 제대로 정하지 않는다.
(6) Business의 문제로 보지 않고 단순히 기술적 차원의 문제로 간주한다.
(7) 외부 전문가의 도움을 구하지 않는다.
(8) 내부 프로그래머의 Career Path를 관리해 주지 않는다.
(9) 협력업체와 전반적인 업무협조체제를 구축하지 않는다.
(10) 타 언어나 시스템, HW까지 전반적인 영향을 고려하지 않는다.

맺는 말

가트너 그룹의 분석에 따르면, 1996년에는 전체 프로그램의 20% 정도가, 그리고 1999년에는 약 90% 정도가 2000년도 문제로 인하여 제대로 작동하지 않을 것으로 예상되고 있다. 더구나 서기 2000년까지는 이제 불과 38개월밖에 남지 않았다. 현재 미국에서는 약 30%의 회사들이 이미 이 문제의 해결을 위하여 착수하였으며, 20%정도가 실시를 검토중이고, 나머지 50%는 아직 대처하지 못하고 있는 것으로 알려지고 있다.

아무도 이 문제를 단숨에 해결하는 마술같은 요술방망이를 당신에게 제공해 줄 수는 없을 것이다. 결국 여러분은 스스로의 노력으로 이 문제를 해결한 다음, 새로운 21세기를 맞이해야 하는 것이다. 그러면 명확하게 해결 시한이 정해져 이는 이 문제를 성공적으로 해결하기 위한 지름길은 무엇일까? 불행하게도 오직 하루라도 빨리 시작하는 것 외에는 별다른 방법이 있을 수 없음에도 불구하고, 아직도 국내 대부분의 전산관리자들이나 프로그래머들은 이 문제의 심각성을 미처 깨닫지 못하고 있는 것 같아 안타깝기 그지없다.