□ 특별기고 □

Java 환경에 대한 고찰

한남대학교 이강수* · 조윤희**

● 목 차 ●

1. 서 론

이미 60년대 말에 연구되었던 인터넷 개념은 25년이 지난 지금 정보통신 하부구조의 발전과 퍼스컴의 홍보에 힘입어 어린이용 인터넷 응용 시스템(즉, 키드넷)까지 등장하여 어린이가 가정에서 인터넷을 접속할 수 있게까지 발전되었다. 또한, 70년대부터 연구되어온 분산 처리 시스템도 최근에 와서 실용화되고 있으며, 80년대부터 연구되어 온 객체지향 기술도 최근 활발히 응용되고 있다.

이러한 상황에서, 70년대와 80년대에 전산학을 공부한 대부분의 전산학자들은 딜레머에 빠지게 되었다. 즉, 이미 70년대에 교과서에도 등장했던 용어와 개념들이 산업계에서 마치 새로운 개념인 것처럼 새로운 용어로서 과대 포장되어 학계로 역류하고 있기 때문이다. 실제로 최근의 신 기술로 알고 연구한 객체지향, 분산 처리, 클라이언트/서버 및 인터넷 기술들은 이미 70년대 및 80년대의 학창시절에 공부했던 내용이었음을 알게된다. 이러한 문제는 전산 분야의 이론과 응용간의 시간 차이 때문이라 생각된다.

본 글에서는 이러한 관점에서 최근 관심을 모으고있는 자바(Java) 환경에 대해 자바 연구자가 사용한 철학(또는, 기본 개념)과 그 철학의 실현 결과(즉, 자바에서의 솔루션)를 검토하고자 한다. 썬마이크로시스템즈사(이하 "썬사"로 약칭함)의 솔루션인 자바 환경은 다른 연구소, 표준기관 또는 벤더 등의 표준안 또는 솔루션들과 매우 유사하다. 본 글은 자바 환경에서 제시한 개념들을 살핌으로서, 전산 분야의 신 조류를 파악할 수 있도록 하는 것이며, 자바 환경에 대한 홍보를 목적으로한 것은 아님을 명시한다. 또한, 본 글은 현재까지 썬사에서 제공한 각종 자료를 바탕으로 연구된 것이며, 썬사의 실제 개념과 다소 차이가 있을 수 있음을 알린다.

본 글의 2장에서는 자바 환경에 관한 일반적인 사항들을 요약하고, 3장에서는 객체지향 언어, 분산 객체 모델, 네트워크 부라우저, 소프트웨어 공학, 분산처리 및 멀티미디어, 네트워

*종신회원
**학생회원

크 컴퓨터 분야의 관점에서 각각 분석한다. 4장에서는 자바에서 사용한 기본 설계의 개념들을 분석하고, 5장에서 결론을 맺는다.

2. 자바 환경 개요

2.1 배경

90년대초 GUI의 개발 경험을 통해 객체지향 기술이 성숙되었고, 인터넷을 통한 웹의 실용화와 보급이 확산되었으며, 개방형 분산 클라이언트/서버 시스템의 장점이 인식되기 시작하였다. 또한, 이러한 환경과 기술을 이용한 웹과 분산 객체 처리 응용 시스템의 개발이 필요해졌고, 이에 대한 솔루션들을 각 벤더들이 제시하였다. 예를들어, 4세대 언어에 의한 RAD (rapid application development)기술, 마이크로소프트사의 분산 OLE(object link & embedding), OSF의 DCE(distributed computing environment), OMG의 CORBA(common object request broker architecture) 등 그것이며, 자바 환경은 썬사의 '통합 솔루션'이라 할 수 있다.

여기서, 통합 솔루션이란 새로운 프로그래밍 언어 및 컴파일러, 개발지원 도구, 보급 방법 및 프로그램의 수행 환경 등을 일괄적으로 제공하므로서, 일관성있게 시스템을 개발 및 운영할 수 있도록 하는 것을 의미한다. 본 글에서 자바 '환경'이란 용어를 사용하는 것도 자바가 이와같이 프로그램 언어 이상의 것을 포함하고 있기 때문이다.

'자바'는 썬사에서 발표한 새로운 프로그램 언어의 이름이기도 하며 남미산 고급 원두 커피의 이름이다[1]. 자바의 로고에 커피잔이 등장하는 것도 이 때문이다. 자바 환경은 91년부터 썬사의 James Gosling의 연구팀에 의해 가전제품 개발 및 사용상의 호완성을 높이기 위해 C++ 언어를 수정한 Oak언어(후에 자바 언어로 명칭을 바꿈)를 정의하므로서 개발이 시작되었으며[1~3], 94년중반부터 폭발적인 관심을 끈 인터넷 웹과 부라우져 개념과 결합하였고, 95년 5월에 성공적인 발표를 끝내고 현재에도 계속 기능을 보강하고 있는 '진행중'인 프로젝트이다[4~18].

2.2 설계 파라다임

자바 언어는 "간단하고, 객체 지향적이며, 분산적이며, 해석가능하며, 강력하고, 보안적이고, 아키텍처와 무관하고, 고성능이며, 멀티 쓰레드 동적 언어"라 정의하고 있으며[1], 분산 시스템의 소프트웨어 개발 개념 및 방법을 대부분 포함하고 있음을 알 수 있다. 표 1은 자바의 설계 목표, 해결 방법 및 효과를 나타내며 그림 1은 자바 환경의 프레임워크를 보인다.

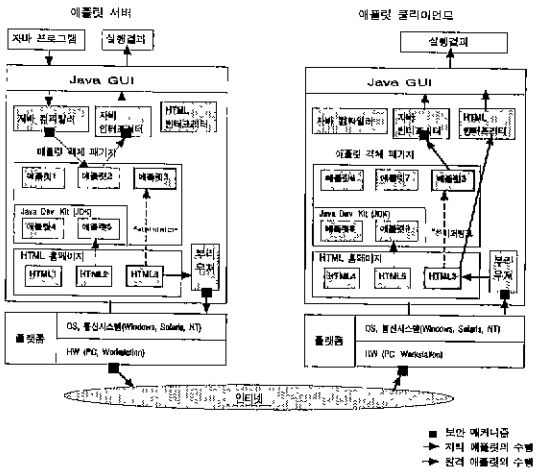

그림 1 자바 환경의 프레임워크

2.3 구성 요소

자바 환경은 다음과 같은 요소로 구성되며 현재 계획되어 있거나 개발중인 것들도 있다.

가. 자바 언어[6] : 새로운 병행(concurrent) 및 객체지향 프로그래밍 언어로서, C · C++ · Objective-C · Modula · Mesa · Modula-3 및 LISP언어들의 특성들을 이용하여 정의하였다. 설계 목표는 표 1에 제시된 자바 환경의 전체 설계 목표와 같으며, 최근의 핵심 개념들을 표현할 수 있다. 그림 3-(a)는 자바 언어의 예를 보인다.

나. 자바 컴파일러와 인터프리터 : 자바 언어의 중간 코드인 자바 바이트 코드('애플릿'이라 한다)로 번역하며(컴파일 기능) 바이트 코드를 실행한다(인터프리터 기능). 이 개념은 파스칼의 P-code개념으로부터 유래되었고 네트워크 상에서 다양한 컴퓨터 플랫폼간의 호환성

표 1 자바 환경의 설계 목표와 해결 방법[1,2,3]

설계 목표	해결 방법	효 과
간략성	-잘 알려진 기존의 언어 선택 (C++) -잘 안 쓰이는, 혼동 가능한, 복잡한 개념들 삭제 (예;연산자 오버로딩, 포인터 연산, 다중 상속 등) -프로그래밍 간략화(예; 자동 가비지 콜렉션)	-새로운 언어에 대한 거부 및 재교육 감소 -분산 소프트웨어의 복잡성 문제 해결 -40KB(라이브러리 포함시 175KB)의 크기
객체지향성	C++ 및 Objective C의 객체 개념 수용	-RAD가능 : 객체(즉, 애플릿) 재사용성, 소프트웨어 IC, Plug&Play
분산성	TCP/IP프로토콜(HTTP, FTP), HTML확장, 웹 부라우저(HotJava, Netscape3.0)	-인터넷 접근 : 개방성, 분산성 -정적(HTML) 및 동적(수행가능) 파일(애플릿) 다운로드 가능
견고성(robust)	-오류의 예방 : 컴파일타임 체크, 명시적 선언, array이용(포인터 제외함), interface구조	-에러의 조기 예방, 안전성, 융통성
보안성(secure)	-4계층의 보안 매커니즘	-네트워크 및 분산환경에 적합 : 바이러스 방지 및 인증성
아키텍쳐 중립성	-Java virtual machine[7] : 바이트 코드(파스칼이나, 4GL에서 사용하는 P-code개념)	호환성 향상, 수행 속도 저하
이식성(portable)	-표준안 수용 : IEEE 754(32비트 플로팅 포인트 연산), ANSI C, POSIX -컴파일러는 자바로 개발, 바이트 코드 인터프리터는 ANSI C언어로 개발	-이식성 높음 : 유닉스, 윈도우즈, 매킨토시 플랫폼
해석성 (interpreted)	가상 코드인 바이트 코드의 인터프리팅에의한 애플릿의 실행	-RAD가능 : 분산 OLE 디버깅 용이, 실행 속도 저하
고성능	-자바 컴파일러에 최적화 기능 포함 -바이트 코드를 실제 머신 코드로 변환해 직접 수행(자바 인터프리터 사용 안함)	바이트 코드의 실행 속도는 머신 코드의 속도와 차이 없음
멀티 쓰레딩	제록스의 Ceder/Mesa시스템의 모니터 개념, 동기화, 멀티쓰레드(Thread, new, synchronized)	실시간성, 반응 시간 향상
동적(dynamic)	interface구조 : Objective C, 인스턴스 변수와 구현 사항을 포함하지 않음 -runtime representation	-유지보수성 향상 : 환경 변화시 클래스(즉, 애플릿)라이브러리 재 컴파일하지 않음 -객체지향 개념의 장점 살림

문제를 해결할 수 있다. 즉, 각 컴퓨터는 자바 언어 또는 C 언어로 프로그램된 자바 인터프리터를 가져다 자체의 자바 컴파일러 또는 C 컴파일러로 컴파일하여 수행가능 코드를 설치하기만 하면 이 기종간의 호환성 문제(즉, 아키텍쳐 중립성)가 다소 해결된다.

다. 자바 애플릿(Applet) : 바이트 코드로 구성된 단위 프로그램 컴포넌트(또는 모듈, 객체)이다. 자바 프로그램을 작성하여 이를 컴파일하므로서 얻거나, 네트워크상의 다른 자바 웹 싸이트로부터 HotJava나 Netscape3.0과 같은 부라우저를 통해 다운로드 받을 수 있다. GUI, 멀티미디어, 입출력 등 자주 쓰이는 애플릿들은 애플릿 패키지 내에서 '상속'받을 수 있

다.

라. 자바 가상 기계(Java Virtual Machine, JVM)[7] : 자바의 바이트 코드가 수행(인터프리트)되는 가상적인 기계로서 일종의 스택 머신 형태이다. 즉, JVM의 어셈블리 코드가 바이트 코드인 셈이다. 또한, 바이트 코드의 집합은 애플릿이 된다. 그림 3-(b)는 바이트 코드의 예를 보인다.

마. 자바 Development Kit(JDK)[5,7,15]: 자바 응용 프로그램 인터페이스(API) 라고도 하며 RAD개념을 통해 자바 환경의 API 및 GUI를 쉽게 개발할 수 있도록 제공된 애플릿 패키지이며 클래스 계층을 이루고 있다. 자바 프로그램 내에서 각 애플릿들을 선언 및 호출하면 수행시간에 클래스 로더에의해 해당 애플릿이 삽입(즉, 링크)된다. 이 개념을 '동적 바인딩'이라 하며 '개방형 서브루틴' 또는 '매크로' 개념과 유사하다. JDK에는 자바 언어 컴파일 지원(java.lan.*), 애플릿 생성지원(java.applet.*), GUI지원(java.awt.*), 유틸리티(java.util.*), 멀티미디어(java.awt.image.*), 입출력(java.io.*), 통신(java.net.*) 등 멀티미디어 정보통신 및 GUI개발에 필요한 대부분의 애플릿들이 있고 계속 그 수가 증가되고 있다.

바. HotJava : Netscape과 유사한 웹 부라우저로서 기존의 HTML문서(문자 및 정지화상)뿐 아니라, 애플릿들도 검색 및 다운로드 받을 수 있다.

사. Object Java 및 OLE Java : 자바 환경을 기존의 IBM이나 마이크로소프트 환경과 인터페이스하기 위한 것으로 현재 개발 중에 있다. Object Java는 자바 애플렛이 무리없이 IBM의 분산 OpenDoc요소와 통합되어 조작할 수 있는 기술이며, OLE Java는 마이크로소프트의 OLE를 지원하며 차기 버젼이 윈도우즈-95에 추가될 것이다.

아. 자바 DB 및 자바 OS : 현재 개발중인 시스템으로서 자바 DB는 자바 환경과 기존의 DB간을 연결하므로서, 클라이언트/서버 데이터베이스를 접근하도록 하는 것이며 인트라넷 시스템개발시 유용할 것이다. 또한, 자바 OS는 자바 터미널을 위한 운영체제로서 간단한 애플릿 관리, 자바터미널내의 메모리관리, 멀티미디어 입·출력, 애플릿 다운로딩, 보안 문제 처리 등의 기능을 수행할 것으로 판단된다.

자. 자바 터미널(또는, 자바 스테이션)[14] : 오라클사에서 제시했던 인터넷 PC 또는 네트워크 컴퓨터(NC)와 유사한 개념이다. 자바 터미널에는 자바 인터프리터와 HotJava 등과 같이 최소한의 자바 환경을 유지하고 전용 자바 칩을 이용하므로서 PC보다 싼 500$ 정도의 컴퓨터를 구성한다는 개념이다. 이미 토스터 크기만한 자바 스테이션 시제품이 개발되었다.

그림 2와 3은 자바 환경에서 자바 프로그램의 실행 과정을 보인다. 특히, 그림 2에서 자바 칩과 자바 OS는 현재 개발 중인 것으로 알려져 있다. 여기서 알 수 있듯이 자바 환경은

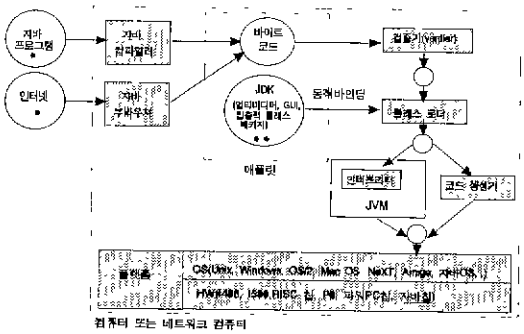

그림 2 자바 프로그램의 수행 과정
(패트리네트형태로 표현함)

```
class vecto                          aload 0          Load this
   int arr[],                        getfield #10     Load this.arr
   int sum[]                         astore 1         Store in la
      int la[]=arr,                  iconst 0
      int S=0,                       istore 2         Store 0 in S
      for (int i=la length  --i>=0;) aload 1          Load la
         S+=la[i]                    arraylength      Get its length
      return S,                      istore 3         Store in i
   }                              A   iinc 3 -1        Subtract 1 from i
}                                     iload 3          Load i
                                      ifLt B           Exit loop if<0
                                      iload 2          Load S
                                      aload 1          Load la
                                      iload 3          Load i
                                      iaload           Load la[i]
                                      iadd             Add in S
                                      istore 2         Store to S
                                      goto A           Do it again
                                   B  iload 2          Load S
                                      ireturn          Return it
```

(a) 자바 프로그램 (b) 바이트 코드

그림 3 자바 프로그램과 바이트 코드의 예[13]

자바 칩부터 GUI개발도구 및 웹 부라우져에 이르기까지 통합적인 솔루션을 제공해 주는 개발 및 운영 환경임을 알 수 있다.

3. 자바 환경의 관점

자바 환경은 통합적 시스템이므로 이를 보는 관점에 따라 그 모양과 평가가 다양할 수 있다. 여기서는 자바환경을 새로운 객체지향 언어, 분산 객체모델, 네트워크 부라우져, 소프트웨어공학, 분산 멀티미디어 및 네트워크 컴퓨터로 간주하므로서 자바 환경의 특성을 살핀다.

3.1 객체지향 프로그래밍 언어

자바 언어는 C++의 아류(dialect) 언어 중의 하나라 볼 수 있으며, 다음과 같은 개념들을 기존의 언어들로부터 도입하였다.

- 기본 구조 및 표현 : ANSI C언어
- 객체지향 개념 : C++
- 인터페이스(Interface) 개념 : Objective-C
- 패키지(Package) 개념 : Modula
- 동시적(concurrency) 개념 : Mesa
- 예외상황(Exception) 개념 : Modula-3
- 동적 링크 및 자동 메모리(dynamic link, automatic storage management) 개념 : LISP

가. 간략성 : 썬사의 RISC(reduced instruction set computer) 아키텍쳐 개념으로부터 유래되었다고 생각되며, 잘 안 쓰며 이해하기 어려우며 보안상(예를 들어, 메모리 violation 등) 문제가될 수 있는 다음과 같은 C 및 C++의 메커니즘들을 삭제하였다[6,16]. 특히, 포인터 연산을 삭제한 것이 돋보인다.

> 다중상속, 포인터 연산, 자동 데이터 형 강제를 지원하는 overloading연산자, 다중 의미 해석을 갖는 연산자, 헤더파일, 전 처리. structure, union, multi-dimensional array, template

나. 객체지향성 : 단일 상속만을 지원하며, 기본적으로 C++의 거의 모든 객체 개념을 포함한다. 특히, Objective C로부터 유래된 'Interface' 구조는 추상적인 클래스 또는 정보 은닉 개념과 같으며, 프로그램 개발시 클래스의 구현(코딩)을 나중으로 미룸으로서 단계적 세련화(stepwise refinement) 개념을 이용할 수 있게 하였다. Modula언어로부터 유래된 'Package' 구조는 유사한 기능을 갖는 에플릿 클래스들의 모임으로서 클래스의 관리와 사용을 용이하게 한다.

다. 동시성 : 자바 언어는 프로그램 수준에서 '멀티쓰레드'를 지원하므로, 한 애플릿 클래스 내에 있는 다수의 매소드들을 다중 처리할 수 있다('new' 명령에의해 새로운 쓰레드가 생성됨). 이때 다중처리 되는 매소드들간의 '동기화'(concurrency) 문제가 발생한다. 자바 언어에서는 동기화 문제를 해결하기 위해, 60년대 말에 한센 및 호어에 의해 제안된 '모니터' 개념을 바탕으로하는 언어인 제록스사의 Mesa언어의 모니터 구조를 채택하고 있다. 모니터는 한 클래스 내에서 동시에 작동되는(즉, 멀티쓰래딩 되는) 매소드의 헤드부분에 'synchronized' 선언을 삽입하므로서 매소드들간의 상호배타적(mutual exclusion) 공유 자료의 접근을 제어할 수 있다. 그러나, 모니터 방법은 공유메모리를 이용한 동기화 방법이므로, 같은 메모리 공간 내에서 작동되는 매소드(즉, 프로세스)들간에만 동기화가 가능하다. 네트워크내의 서로 다른 시스템에서 동시에 작동되는 에플릿 객체들간의 동기화에 관한 솔루션은 OSF의 DCE에서 사용하는 RPC(Remote Procedure Call)을 이용할 것으로 알려져 있다.

라. 견고성 및 보안성 : 자바 언어는 컴파일시 엄격한 자료 형 검사, 명시적 매소드 선언, 포인터 연산 배제, 자동 가비지 콜렉션, 미리 인터페이스 정의(즉, 빠른 프로토타이핑이 용이하며 컴파일시 오류 발견기회가 높아짐), 자바 가상 기계(즉, JVM, 바이트 코드에 의한 인터프리팅) 및 예외상황 처리 메커니즘(즉, C++와같이 throw, try 및 catch구문 이용) 등 프로그램의 견고성과 보안성을 높이기 위한 메커니즘들이 포함되어 있다. 특히, 자바는 네트워크 환경에서 운영되므로 보안성을 높이기 위해 다음과 같은 4가지 수준의 보안 메커니즘[13]들이 돋보인다.

- 수준 1 : '자바 언어' 자체와 '자바 컴파일러'에서 이루어진다. 예를 들어, 포인터 사용 금지, 애플릿의 시스템 힙(heap), 스택 및 메모리 접근 금지 등이 있다.
- 수준 2 : 'Verifier'가 바이트 코드(즉, 애플릿)의 검증 기능(바이러스 체크 등)을 수행한다.
- 수준 3 : '클래스 로더'가 메소드, 파일, 디렉토리 접근 제어 및 방화벽(firewell) 기능을 수행한다.
- 수준 4 : 네트워크 상에서 애플릿 파일의 안전한(secure) 전달 및 보호(protection)를 위해 공개키 방식의 RSA(Rivest-Shamir-Adleman) 암호 알고리즘을 이용한 디지털 서명, 사용자 인증(authentication) 및 비밀성(secrecy) 보장 서비스 등을 제공하고 있다. 네트워크 상에서의 보안 메커니즘인 PGP(Pretty Good Privacy)나 PEM(Privacy Enhanced Mail) 등을 활용할 것으로 생각된다.

표 2 자바 언어의 비교[6]

특성 / 언어	Java	Small Talk	TCL	Perl	Shell	C	C++
간략성	A	A	A	B	B	B	C
객체지향성	A	A	C	A	C	C	B
견고성	A	A	A	A	A	C	C
보안성	A	B	B	A	B	C	C
인터프리팅	A	A	A	A	A	C	C
동적성능	A	A	A	A	B	C	C
이식성	A	B	A	A	B	B	B
플랫성	A	B	B	A	B	C	C
멀티스레딩	A	C	C	A	C	C	C
커버지 콜렉션	A	A	C	C	C	C	C
예외상황 처리	A	A	C	C	C	C	B
성능	A	B	C	B	C	A	A

마. 기타 특성 : 자바 언어는 모든 객체의 형(type)을 미리 정의하므로서, 컴파일시 형의 체크가 가능하다. 또한, 바이트 코드를 사용하고, 이름(name)에 의한 메소드 및 인스턴스 변수를 참조하게 하고, IEEE745(즉, 32비트 정수 및 64비트 부동 소수점 표현 방법의 표준안)와 POSIX 표준을 수용하고, 자바 인터프리터('실행시간 환경'이라 함) 및 인터페이스를 ANSI C언어로 구현하므로서, 아키텍쳐 중립성을 높였다. 표 2는 자바 언어와 다른 언어들과의 차이를 보인다.

3.2 분산 객체 모델

자바 환경은 최근 시스템의 특성인 분산형, 개방형 및 객체지향 구조를 설계 목표로 하였고, 분산 객체 시스템의 국제 표준 모델인 OMG의 CORBA모델에 대한 썬 사의 솔루션 중의 하나라고 볼 수 있다. 즉, 자바 언어의 'Interface'는 CORBA의 IDL(Interface Description Language) 인터페이스와 매우 유사하므로, IDL을 자바 언어로 변환하므로서 CORBA를 자바 환경에서 구현할 수 있다. CORBA의 IDL, 응용객체, 공통 지원기능/객체 서비스 및 객체 요청 중재자 (ORB)는 각각 자바 환경에서 자바 언어, 응용 애플릿, 애플릿 클래스 패키지(JDK, API) 및 HotJava에 각각 대응할 수 있다. 그러나, ORB에 해당하는 HotJava는 아직 그 기능이 매우 부족하며 이에 대한 보강이 계속 될 것으로 생각된다.

3.3 네트워크 부라우저

자바 환경(특히, HotJava)은 인터넷상의 새로운 웹 부라우저이다. 즉, Mosaic이나 Netscape와 같은 부라우저들은 HTML이나 VRML로 작성되고 HTTP나 FTP프로토콜 등으로 송수신되는 문서 및 이미지 파일을 다운로드하고 있지만, 자바 환경에서는 문서나 이미지 파일뿐 아니라 프로그램(즉, 애플릿)을 다운로드하여 자신의 플랫폼 상에서 즉시 수행할 수 있다. 이 방법은 홈페이지에 있는 에니메이션 화상을 화상이 아닌 '화상을 생성하는 애플릿'을 다운로드하여 자신의 컴퓨터에서 실행하므로서 통신량이 절약된다는 장점을 가지고 있다. 또한, 소프트웨어 부품(예를 들어, 수학 함수, 스프래드쉬트 함수 등) 개념이 실현된다. 그러나, 애플릿은 동적인(실행 가능한) 정보이므로, '논리적 폭탄(logical bomb)' 또는 '트로이 목마' 형 '바이러스 애플릿'이 인터넷을 통해 급격히 확산될 가능성이 있다. 이 문제는 자바의 개발 초기부터 염두에 두고 있으며, 앞 절에서 논한 4가지 수준의 보안 메커니즘을 이용하므로서 다소 해결할 수 있다.

3.4 소프트웨어 공학

자바 환경은 RAD 또는 rapid prototyping

을 위한 통합 지원 도구(즉, I-CASE)로 간주할 수 있으며, 애플릿(즉, 객체화, 재사용 및 상속), Plug & Play, GUI개념 및 JDK패키지 등을 활용한다. 4세대 언어로 알려진 Visual Basic, Power Builder, Delphi등과 마이크로소프트의 분산 OLE같은 RAD도구들은 자바처럼 새로운 언어에서 웹 부라우져까지의 통합 환경을 제공하지는 못하고 있다.

3.5 분산 처리 및 멀티미디어 시스템

자바 환경은 클라이언트/서버 컴퓨팅 모델들 중 '분산 응용(distributed application)'의 수준으로서, 복잡한 수준에 해당한다. 분산 응용이란 응용 프로그램(애플릿)들을 각 클라이언트와 서버에 분산하여 협동 처리하는 것이다. 이기종간의 협동 처리를 지원하기 위해 OSF의 DCE와 같은 미들웨어 및 그룹웨어가 필요하며 자바 환경이 이들 역할을 할 수 있다.

자바 환경은 분산 멀티미디어 시스템의 해결책중의 하나이다. 애니메이션과 같은 동화상은 에니메이션용 애플릿을 다운로드하여 자체 시스템에서 동화상을 생성하므로서, 실시간 사용자 개입이 가능하다. 그러나, 영화와 같이 프로그램화 할 수 없는 동화상들은 현재의 방법을 이용하여 각 프레임별로 다운로드하여 동화상을 재생할 수 있다. 즉, 자바는 VOD(Video On Demand) 시스템의 '부분적인' 해결방법에 지나지 않는다.

3.6 네트워크 컴퓨터

자바 환경은 새로운 네트워크 컴퓨터(NC)의 구조라 할 수 있다. 오라클사의 인터넷 PC와 같이 자바 터미널 개념은 인터넷을 활용한 경제적인 컴퓨터의 활용방법이다. 즉, 각 NC에는 최소의 하드웨어와 소프트웨어를 가지며 필요시 인터넷을 통해 프로그램을 다운로드하여 수행한다는 개념이다. 자바 터미널의 경우 개발중에있는 25$ 정도의 자바 칩(JVM을 하드웨어로 구현한 것)과 최소한의 RAM이 내장된 마더보드에 저 용량의 하드디스크, 통신카드, 간단한 OS(즉, 자바 OS) 및 멀티미디어 카드로 구성될 것으로 보인다[14].

4. 자바 환경의 특징 및 활용 가능성

4.1 특징

자바 환경이 관심을 불러일으킨 것은 다음과 같은 이유에서라 판단된다.

가. 마케팅의 성공 : 자바 환경에 관심이 높아진 것은, 최근의 PC의 다량 보급, PC에 의한 일반인들의 인터넷 접속 및 웹의 검색 기회 확대, 객체 지향 분석 및 설계 기술에 대한 관심의 확대 및 분산 시스템에 대한 활발한 연구 및 개발 등의 '시대적 상황', 썬 사의 대대적인 '마케팅' 및 '언론사'들의 정보화 사회에 대한 분위기 조성 사업의 3가지 요소가 일치했기 때문이라 판단된다. 이는 1995년도에 마이크로소프트사가 이루웠던 윈도우즈-95에 대한 마케팅의 성공과 같은 사례이다.

프로그래밍 언어, 운영체제 및 컴퓨터 시스템이 관심을 끌고 지속적으로 사용되기 위해서는 그 성능의 우수성에 못지 않게 정부기관 또는 대기업의 지원이 중요하다. 프로그래밍 언어들중 FORTRAN, COBOL, PL/1, C 및 Ada가 대표적인 경우이다. 또한, 60년대 중반에 발표한 MULTICS는 성능면에서 매우 우수한 운영체제였지만, 마케팅에 실패하므로서 당시 경쟁 운영체제였던 IBM의 OS/360에게 운영체제의 대명사 자리를 빼앗긴 바 있다(사실, "OS"는 IBM의 운영체제 이름이다). 자바 환경의 경우, 지금까지는 마케팅과 분위기 조성에 성공한 것으로 판단된다.

나. 통합적 솔루션 제공 : 자바 환경은 프로그램언어부터 웹 부라우져까지 컴퓨터의 모든 계층(즉, 하드웨어, 시스템 프로그램, 미들웨어 및 응용 프로그램 계층)에대한 일관적인 통합 솔루션을 제공하고 있다. 이는 지금까지 각 벤더들이 제시한 솔루션들과는 다른 것이다.

다. 간단한 개념 : MULTICS는 '스위스 군용칼' 처럼 기능이 다양하지만 사용하기가 복잡했었던 반면, 그 일부 기능(예를들어, 파일 시스템과 커널)만을 포함한 운영체제인 유닉스나 MS-DOS/윈도우즈는 간단했기 때문에 성공한 것으로 평가된다. 그러나, 유닉스가 다시 MULTICS만큼이나 복잡해져가고 있으며, 유닉스의 장점이 줄어들므로 윈도우즈-NT 등의

도전을 받고있다. 60년대 말의 "소프트웨어 위기"가 재현되는 느낌이다. 자바의 개발자들은 이러한 문제를 인식하여 C++언어(MULTICS나 Algol68과 비유된다)의 복잡한 개념들을 과감히 없앤 언어인 자바언어(Unix나 파스칼과 비유된다)를 제시하였다. 즉, "Simple is beautiful." 철학을 따르고 있다.

라. 전통적인 기술과 신 기술의 조합 : 사실 자바 환경은 새로운 기술을 이용한 것은 아니다. 예를 들어, 바이트 코드 개념은 이미 1970년대초 파스칼 컴파일러의 구현시에 성공적으로 사용되었으며, 객체 개념도 80년대 초부터 사용되기 시작하였다. 자바 환경은 혁신적인 방법보다는 이미 잘 알려져 있고 친숙한 기존의 방법들을 유기적으로 결합시켰다. 즉, "Old is, but good is." 철학을 따르고 있다.

4.2 응용 시나리오

자바 환경은 다음과 같은 두 가지 가상적인 시나리오를 통해 그 개념과 응용 가능성을 이해할 수 있다.

가. 멀티미디어 프로그램 부품의 통신판매

- 자바 애플릿(또는, 프로그램, 객체, 컴포넌트)의 제삼자(Third Party)업체인 JBA사는 게임용 에니메이션 애플릿들을 자바 언어와 JDK를 이용하여 코딩하고 자바 컴파일러로 컴파일하여 바이트 코드 상태의 파일을 만든다. JBA사는 자사의 홈페이지를 구성하고있는 HTML파일에 문자 및 정지화상 파일과 개발한 애플릿들을 포함시킴으로서, 애플릿들을 통신 판매할 준비를 완료한다.
- 멋있는 게임 시나리오를 생각하고 있던 G군은 게임 프로그램을 자바 언어로 개발하기로 결심하고 게임에 필요한 각종 에니메이션 애플릿을 구입하기 위해 Netscape3.0이나 HotJava를 통해 인터넷의 웹들을 항해하기 시작하였다. G군은 마침내 JBA사의 홈페이지로부터 자신이 원하던 애플릿이 있음을 확인하고 이를 자신의 컴퓨터로 다운로드 받았다. 이때, 자바에 있던 인증 서비스(전자서명)를 이용하여 라이센스를 취득하고 그 비용을 전자결제 하였다.
- G군은 구입한 애플릿을 자신이 개발하고 있던 게임 프로그램에 무리 없이 Plug & Play방법으로 설치하여 게임 프로그램을 완성하였다. 완성된 프로그램은 바이트 코드 형태이므로, 자바 인터프리터를 이용하여 자신의 PC 586 윈도우즈-95 플랫폼에서 실행시킬 수 있었다. 몇달 후 자바 칩이 내장된 네트워크 컴퓨터를 구입하여 보다 빠르게 게임을 즐길 수 있었다.

나. 분산처리 시스템 개발

- 100개의 클라이언트를 갖는 분산 클라이언트/서버 시스템을 개발하기로한 JAB사에는 100명의 개발 팀이 각 지사별로 흩어져서 근무하고 있고, 각 개발자들은 자신의 PC또는 WS들이 인터넷에 연결되어있다. 100개의 클라이언트는 유닉스 및 윈도우즈 등 여러 종류의 플랫폼으로 구성되었다.
- 서로 다른 위치에 있는 100명의 개발자는 100여개의 클라이언트들에 설치할 GUI용 애플릿들을 나누어 동시에 개발(자바 프로그램 작성 및 컴파일)하여 바이트 코드 상태로 통합 시험용 컴퓨터로 송신한다. 애플릿들은 JDK와 RAD개념을 이용하므로서 쉽게 개발할 수 있었다. GUI용 애플릿들을 매우 쉽게 통합 시험한 후 이를 복사하여 100개의 클라이언트에 송신하였다.
- 시스템의 응용 프로그램(애플릿)들도 100명의 개발자가 동시에 개발하여 한곳에서 통합 시험을 완료하였다. 완전한 분산 시스템을 위해 각 응용프로그램은 분할하여 100개의 클라이언트 시스템에 나누어 송신하였다.
- 각 클라이언트에서는 자신의 플랫폼에서 수신된 GUI와 응용 애플릿을 자바 인터프리터를 실행하므로서 완전한 분산 처리가 가능하게 되었다. 데이터베이스 서버도 자바 DB를 이용하므로서 매우쉽게

개발할 수 있었다.

- 시스템의 환경이 바뀌어 GUI나 응용프로그램을 교체할 때도 교체할 애플릿을 인터넷을 통해 각 클라이언트로 브로드케스트함으로서 일시에 교체가 되고 100곳이나 되는 클라이언트마다 설치해야하는 번거로움이 없어졌다. 각 클라이언트의 플랫폼이 바뀌어도 GUI나 응용 프로그램이 바이트 코드 형태의 애플릿으로 구성되어 있음으로 ANSI C언어로된 자바 인터프리터만 다시 컴파일하므로서 시스템의 유지보수 비용이 절감되었다.

5. 결 론

지금까지 살펴본바와 같이 자바 환경은 비교적 우수한 것으로 판단된다. 한가지 아쉬운 점은 자바언어를 가독성(readability)이 낮고 준고급언어인 C나 C++를 기반으로 하여 만들었으나, 파스칼·포트란 및 베이직 등 간단한 고전적인 언어에 기초를 두고 만들었다면 현재의 자바 언어보다 가독성, 구조성 등이 높아졌을 것이라는 점이다. 어차피, 자바 언어는 C 언어의 특성인 포인터를 허용하지 않기 때문이다.

자바 환경도 유닉스나 윈도우즈-95처럼 장점이 잘 홍보된 상품에 지나지 않는다. IBM OS, 유닉스 및 윈도우즈가 일개 밴더의 상품이긴 했지만, 컴퓨터 산업과 학문에 끼친 영향은 무시할 수 없다. 현실적으로, 기업에서는 전산학을 연구한 학생보다는, 유닉스, 윈도우즈 프로그래밍, Visual Basic 및 Powerbuilder를 잘 쓰는(?) 학생을 요구하기 때문이다. 국내 연구자나 밴더들은 자바 환경으로부터 기존의 전산분야의 문제점들을 파악하여 이를 통합적으로 해결하고 마케팅까지 성공시키는 방법을 배울만하다. 따라서, 자바 환경에 대한 학문으로서의 관심과 연구가 필요하다.

참고문헌

[1] A. van Hoff, S. Shaio, O. Starbuck, "Hooked on Java", Addison Wisley, 1996.

[2] E. Tittel, M.Gaither (홍우택 역). "깡통을 을위한 Java", 비엔씨, 1996.

[3] John December (홍우택 역), "자바 맛보기", 비엔씨, 1996.

[4] E. Anuff, "JAVA Source Book", John Wiley & Sons, 1996

[5] L. Lemay, C. L. Perkins, "Teach Yourself JAVA in 21 Days", Sams Net Pub., 1996.

[6] James Gosling, Henry McGilton, "The Java Language Environment : A White Paper", Version 1.0 Beta, Sun Microsystems, Oct 30, 1995.

[7] "The Java Virtual Machine Specification", Release 1.0 Beta, Sun Microsystems, Aug. 21, 1995.

[8] 신명기, "자바가 세상을 바꾼다", 마이크로소프트웨어, 1996년 1월호, pp.263-275.

[9] 최희창, "자바의 응용과 시스템 프로그래밍", 마이크로소프트웨어, 1996년 1월호, pp.276-289.

[10] 박경순, "자바와 신 지구촌 인터넷의 미래", 마이크로소프트웨어, 1996년 1월호. pp.290-297.

[11] "한글 Java Frequently Asked Question (FAQ)", http : //www..nanum.co.kr/pub/javafaq.html

[12] "Sun Java Day"자료, 여의도 63빌딩, 한국썬.마이크로 시스템즈, 1996.3

[13] Papers down loaded from Java Home Pages, http://java.sun.com, http://www.sun.com, http://www.javasoft. com

[14] "한국 썬 소식", 한국썬.마이크로시스템즈. 1995년 6월, 1996년 1~3월호.

[15] JAVA API User's Guide, 1996.

[16] 조윤희, 이강수, "Java와 C++의 비교연구". 한국정보처리학회 학술발표 논문집, 부산수산대, 1996년 4월.

[17] "특집 : 자바 애플릿의 향기", 프로그램 세계, 1996년 4월호, pp115-177.

[18] "JAVA : 대화식 웹 구축엔 최적", 인터넷 지구촌, 하이테크 정보 부족, 1996년 4월 20일.

이 강 수

1981 홍익대학교 전자계산학과 졸업
1983 서울대학교 계산통계학과 이학석사
1989 서울대학교 계산통계학과 이학박사
1985～1987 대전산업대학교 전자계산학과 전임강사
1987～현재 한남대학교 전자계산공학과 부교수
1992～1993 미국 일리노이대학교 객원교수
1995 한국전자통신연구소 초빙연구원

관심분야 : 소프트웨어공학(신뢰도, 프로젝트 관리, 분산 시스템 모델링), 페트리넷(응용), 멀티미디어(동기 모형), 보안 시스템(프로토콜 모델링 및 보안 감사), 자바 환경 등

조 윤 희

1995 한남대학교 수학과 졸업
1995～현재 한남대학교 전자계산공학과 석사과정 재학중

관심분야 : 소프트웨어공학, 페트리넷, 자바, 윈도우즈 프로그래밍 등

●'96 춘계 전산교육 워크숍●

- 일　　자 : 1996년 6월 14일
- 장　　소 : 한국교원대학교 교양학관 1층
- 주　　제 : 교육 세계화에 따른 컴퓨터교육
- 주　　최 : 전산교육연구회
- 문 의 처 : 한국교원대학교 컴퓨터교육학과 김성식 교수
  T. 0431-230-3740