해외안테나

자바는 결코 요술 방망이가 아니다

비전 과대포장, 건강한 회의론 유지 · 검증작업 필요

업계 전반에서 자바라는 말이 빈번히 사용되면서 자바 문제는 모두 해결된 것처럼 생각될 수도 있다. 또 혹자는 모든 기업들이 자바를 갈구하고 있으며, 자바 애플리케이션들이 이제 막 쏟아져 나올 것이라고 생각하고 있을 지도 모른다.

최근 애플리케이션 개발 매니저회의에 참가한 사람들을 대상으로 자바 이용실태를 조사했다. 조사 결과 소수만이 일부 자바 애플리케이션 개발을 테스트하고 있는 것으로 나타났다. 또 2, 3개 기업들이 자사의 개발자들에게 자바 실험을 덤으로 허락하고 있었는데, 주로 개발자들에게 교육기회 및 동기를 부여하기 위한 방법으로 행해지고 있었다.

누가 자바를 요구하는가

그렇다면 자바에 대한 끊임없는 요구는 도대체 어디에 나오는 것인가? 우리는 자바에 대한 이야기를 주로 산업 전문지와 트레이드 쇼, 그리고 순회 회의 같은 곳에서 듣고 있지만 일반적인 기업 애플리케이션 개발 세미나에서는 별로 들을 수가 없다.

이제는 더이상 현실을 왜곡해 받아들이지 말아야 할 것이다. 필자 역시 자바가 거대한 잠재력을 갖는다고 믿고 있는 사람중의 하나이지만 컴퓨터 언어 및 소프트웨어 기반구조가 진화되어가는 현 시점에서 자바는 아직 우세하다고 볼 수 없다. 필자의 소프트웨어 산업 경험에 비추어 볼 때 소프트웨어는 우리가 예상하는 형태로 또는 예상하는 시간에 성공한 적이 별로 없다.

지금으로부터 약 7, 8년 전에 많이 얘기되었던 다양한 개발툴들이 서로 정보를 교환하게 해주는 소프트웨어 백플레인에 대해 생각해보자. 그 당시 제안되었던 소프트웨어 백플레인들은 전혀 성공하지 못한 반면, 툴과 소프트웨어의 통합은 이제 상당히 일반화되었다.

마찬가지로 케이스는 한때 유행어였는데, 그것은 당시의 기술이 지나치게 경직되었기 때문이었다. 그러나 오늘날 개발툴 벤더들은 케이스의 핵심요소인 분석과 설계 및 모델링을 통합하기 위해 전력을 다하고 있다. 이 두가지 경우 모두 구현에는 실패했지만 그 개념은 올바른 것이었음을 입증하고 있다.

자바의 경우에도 이와 유사한 상황이 벌어질 것 같다. 자바는 우리가 알고 있는 것처럼 인생을 변화시키지는 않을 것이다. 자바는 모든 다른 프로그래밍 언어를 밀어내지 않을 것이다. 자바가 애플릿들을 위한 사실상의 언어가 되지도 않을 것이다. 또 모든 하드웨어 플랫폼들을 별로 차이가 나지 않는 상품들로 바꿔놓지도 않을 것이다.

명백한 것은 자바는 처음부터 분산 소프트웨어 환경으로 설계되었다는 점이다. 자바는 구현된 분산 애플리케이션들을 위한 가장 중요한 서버-지향적 언어가 될 것이다. 클라이언트 지향적인 자바 애플릿들은 장기적으로 볼 때 별로 빛을 보지 못할 것이다.

물론 개발자들은 다른 언어를 가지고도 분산 애플리케이션들을 개발해 왔다. 이들과의 차이점은 자바가 분산 아키텍처를 지원하도록 설계되었다는 점이다. 애플리케이션들의 부분들이 수십개의 플랫폼 상에 위치하고 있다면 자바는 각 지역 시스템에 없는 부분들을 찾기만 하면 된다.

따라서 전세계에 걸쳐 사무실을 운영하고 있는 기업을 위한 애플리케이션을 개발할 때 자바를 이용하면 보다 쉽게 할 수 있다.

자바는 분산 환경을 가로지르는 보안, 실행시간 휴대성, 그리고 메시지들과 정보가 환경 전체를 통해서 전해진다는 것을 전제로 하는 에이전트 언어를 제공한다.

요컨대, 자바는 애플리케이션들이 네트웍 전체에 걸쳐 흩어져 있는 것을 예상하고 있다. 자바는 개발자들이 높은 수준으로 추상된 상태에서 애플리케이션을 개발할 수 있도록 해 줄 것이며, 밀접하게 정의된 객체 모델을 전제로 할 수 있게 해 줄 것이다.

C, C++ 또는 다른 언어로 작업하는 개발자들도 분산 애플리케이션을 구축할 수 있다. 그러나 그들은 처음부터 메시지 전달과 통신을 구축함으로써 네트웍 인식기능을 코딩해야 한다.

이와는 대조적으로 자바는 어렵고 복잡한 네트웍 지원 애플리케이션 구축작업의 상당부분을 개발자들의 개입없이 가능하게 해준다.

표준 마련 절실

이러한 자바의 모든 장점과 혜택에도 불구하고, 개발 매니저들은 현명하게도 자바에 대한 신중한 접근태도를 취하고 있다. 특히 주의해야 할 것은 자바에 대한 엄청난 과대포장이다. 벤더들이 자사의 소프트웨어가 기업 문제들을 해결해주는 정교한 제품임을 설득하지 않고 자바로 쓴 제품이라는 것을 강조해 설명함으로써 제품의 가치를 높이려할 때 매니저들이 회의적인 반응을 보이는 것은 어쩌면 당연하다.

왜냐하면 자바로 쓴 소프트웨어라고 해서 기업 문제를 효과적으로 해결하는 것을 보장하는 것은 아니기 때문이다. 자바 가상 머신을 작동하는 모든 플랫폼으로 이식될 수 있는 애플리케이션이 반드시 기업의 성공을 보장하는 것은 아니다.

자바가 실제로 기업에 가치있는 것임을 입증하려면 풀어야 할 과제가 아직 많이 남아 있다. 우선 자바가 이식성을 널리 인정받으려면 표준 구현을 엄격하게 준수해야 한다. 자바가 보편적인 이식성을 약속한 최초의 애플리케이션 프로그래밍 언어는 아니며, C, C++, PL/1, Ada, 그리고 심지어는 코볼까지도 한때는 다중 플랫폼을 가로지르는 휴대가능성을 약속했었다.

C와 C++는 휴대가능성을 상당한 범위로 구현하였으나 그 비용이 현실적인 문제였다. 이 문제를 자바는 자동 쓰레기 수집, 가상 메모리, 그리고 다른 기능들을 통해서 해결해가고 있다.

자바는 또한 플랫폼 이식성을 위해 가상 머신의 존재에 의존하는 첫 프로그래밍 언어도 아니다. 매직 소프트웨어의 매직은 광범위한 기록후 임의배치(write once/deployment anywhere) 이식성을 전달하는 가상 머신을 오랫동안 사용해 왔다. 상당수의 다른 크로스-플랫폼 4GL들도 가상 머신들을 이용하고 있다.

해석된(interpreted) 코드의 성능한계를 고려할 때 순수한 가상 머신 접근방법이 과연 자바를 위한 최선의 선택인지 필자 스스로 확신을 못하고 있다. 그 대신 개발자들은 자바로 애플리케이션을 만든 후 설치를 위해 C 또는 C++로 컴파일하는 것을

더 선호할 수도 있다. 이것은 애플리케이션들이 컴포넌트 수준의 이식성이라는 혜택을 받을 수 있게 해주지만 여전히 컴파일된 코드로서 성능한계를 겪게 된다.

유닉스의 경험을 반추해보면 자바의 미래에 대해 일부나마 통찰할 수 있다. 유닉스는 운영체제 수준의 이식성을 약속했지만 결코 성취한 적은 없다. 대신에 벤더들은 특정한 플랫폼이나 특정 기능성을 제공하기 위하여 성능을 개선한다는 건전한 이유로 유닉스를 확장하고 향상시켰다.

그 결과 많은 서로 다른 유닉스들이 생겨나 이식성에 제약을 받게 되었다. 표준 구현에 대한 엄격한 준수가 없다면 자바 역시 유닉스의 전철을 밟게 될 것이고, 수많은 벤더들과 경쟁하는 집단들로 분할될 것이다.

필자는 표준적인 자바 구현을 원했지만 최근 발표되고 있는 100% 순수 자바 계획에 대해서는 회의적이다. 100% 순수한 자바라는 것이 애플리케이션 개발 용어로서 도대체 무엇을 의미하는지조차 알 수가 없다. '순수한' 이라는 것이 지나치게 엄격하게 되는 것도 걱정스럽다. 순수성이 요구되는 것이 아니라 표준이 요구되는 것이다. 우리는 상호운영성을 보장하기 위하여 일관성있는 인터페이스들이 필요한 것이다.

기술의 공존성

오늘날의 자바는 아직 초기단계에 있기 때문에 홍역을 앓고 있다. 이 언어는 탄생한 지 겨우 2년밖에 되지 않았고, 자바 컴포넌트 상호운영성 표준인 자바 빈즈는 발표된 지 겨우 몇달 밖에 되지 않았다. 비록 다른 벤더들도 준비를 하겠지만 현재로서는 썬마이크로시스템즈의 핫자바(HotJava)라는 단 한개의 브라우저만이 자바 최신 버전을 지원하고 있다.

앞으로 우리는 자바가 급속하게 성장하는 것을 보게 될 것이다. 적시의 컴파일러들은 속도가 증가할 것이고, 애플리케이션 테스트 툴들이 출현할 것이며, 비주얼 통합 개발 환경들은 개선되고 있다. 또한 광범위한 자바 클래스 라이브러리들을 사용할 수 있게 될 것이며, 자바 애플리케이션들을 다른 분산 컴퓨팅 환경으로 연결하는 다양한 미들웨어가 나타나게 될 것이다.

IBM, 썬, 그리고 다른 벤더들은 자바의 연구 개발에 수백만 달러를 쏟아 붓고 있다. 앞으로 3, 5년 내에 이 모든 것들이 자리를 잡게 되면 우리는 마침내 자바를 진지한 서버-지향적인 분산 애플리케이션들과 기반구조들을 위한 하나의 플랫폼으로 고려할 수 있게 될 것이다.

물론 이 모든 것들이 반드시 일어난다는 보장은 없으며, 따라서 오늘날 자바를 선택하는 것은 미래에 대한 도박을 하는 것과 같다. 자바가 어떤 형식으로든 존재할 것이며, 미래의 기업 컴퓨팅에 중요한 역할을 할 것이라는 점은 확신하지만, 자바가 우리가 해오던 모든 것을 바꾸어 놓을 것이라고 생각하지는 않는다.

컴퓨팅의 역사를 돌아보면 하나의 기술이 끝나는 동시에 다른 기술이 시작하는 급격한 중단은 발견할 수 없다. 그 대신에 우리는 오래된 기술이 변화하고 새로운 기술에 직면하여 순응하는 점진적인 진화를 보게 된다. 오늘날 메인프레임, 미니컴퓨터, 스탠드얼론 PC, 랜, 클라이언트/서버 컴퓨팅, 인터넷, 인트라넷 등 모든 것들이 단절되기보다는 서로 다른 기종으로 분산된 환경에서 공존하고 있다.

지금까지의 개발환경과 단절하지 않고도 자바의 미래를 준비할 수 있다. 자바를 배우고 자바를 실험할 수도 있다. 이식성과 분산 객체 접근방법면에서 장점을 가진 프로젝트를 선정해서 시작해 보는 것이다.

이와 동시에 건강한 회의론을 유지해야 한다. 많은 벤더들이 전문적인 유행어를 만들어 고객을 현혹시킬 것이 아니라 그 기술의 사업적 가치를 시연하게 해야 한다. 벤더들에게 그들의 소프트웨어는 어떤 기능을 가지고 있는지, 어떻게 작동하는지, 그리고 어떠한 사업 가치를 제공하는지에 관해 상세한 정보를 제공할 것을 요구해야 한다. 자바를 비롯한 새로운 기술에 운명을 걸기 전에 이러한 검증작업은 반드시 필요하다. DC

〈DBMS U.S.A〉