스크립트 언어의 동향 및 응용 방안

이화여자대학교 김익순

1. 서 론

최근 프로그래밍 언어 기술의 발전과 함께 다양한 형태의 프로그래밍 언어들이 빠른 속도로 등장하고 있다. 이러한 언어들은 객체지향 언어, 명령형(imperative) 언어, 함수 언어, 병렬 언어와 같이 여러 다양한 범주로 분류되어 그 특성들이 이해되기도 한다. 이들 중 특히, 스크립트 언어라고 언급되는 TCL, PERL, PYTHON과 같은 언어들의 동향 및 그 응용에 대해서 살펴보고자 한다.

2. 스크립트 언어의 발전

개발될 소프트웨어의 성격에 따라서 적절한 프로그래밍 언어가 선택되어져야 한다. 적합한 프로그래밍 언어를 선택하기 위하여 많은 사항들을 고려해야 하지만, 반드시 확인되어져야 할 사항으로 소프트웨어의 효율성(즉, 최대 수행 속도와 실행시 요구되는 최소 자원의 추구)과 개발 생산성이 있다. 소프트웨어의 효율성과 개발 생산성, 이 두 사항은 서로 상충되는 면을 가지고 있어서 어느 한쪽만을 강조할 경우, 다른 쪽은 무시될 수 있다. 효율성과 생산성을 모두 고려하여, 절충된 적절한 프로그래밍 언어를 선택하는 것이 바람직하다.

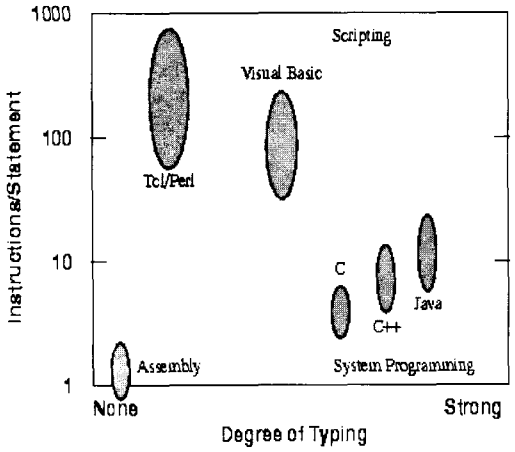

그림 1 각 언어의 표현력 비교자료. 각 언어의 statement 수행에 필요한 어셈블리 코드의 라인수[1]

과거 하드웨어의 성능이 열악하던 시절, 그러한 하드웨어 환경에서 수행될 소프트웨어에게 무엇보다도 요구되었던 것은 바로 효율성이었다. 개발 생산성이 결코 고려되지 않았던 것은 아니지만, 효율성을 떨어뜨리면서 생산성을 강조할 수 있는 상황은 아니었다. 따라서 개발 언어는 생산성은 떨어지더라도 최고의 효율을 얻을 수 있는 언어(가령, 어셈블리어, Fortran, 또는 C/C++)가 선택되었다. 개발하는 소프트웨어가 과연 원하는 정도의 효율성을 나타낼 수 있을지 여부가 최대 관심사였다. 하지만, 하드웨어의 비약적인 발전과 프로그래밍 언어 기술의 발전은 상황을 조금씩 변화시켰다. 아직도 소프트웨어 효율성은 무시할 수 없이 중요하지만, 다른 요인들(제한된 개발 시간, 적은 개발 인원, 소프트웨어 테스팅 및 유지 보수의 어려움 등) 역시 중요한 고려 사항으로 떠오르기 시작했고, 이 과정에서 개발 생산성은 결코 무시할 수 없는 사항이 되었다. 개발 생산성을 높이기 위한 많은 시도가 있었고, 이러한 추세는 프로그래밍 언어에도 영향을 미치기 시작했다. 이러한 과정에서 많은 스크립트 언어들이 새로이 등장하기 시작했다. 스크립트 언어들은 개발 생산성이 높고 표현력이 극히 우수한 것으로 평가된다[1].

3. 스크립트 언어의 특징

(우수한 표현력) TCL, PERL, PYTHON과 같은

스크립트 언어들은 범용 프로그래밍 언어로서의 모습을 갖추고 있으며 우수한 표현력을 특징으로 생산성 향상을 꾀하고 있다. 그림 1에 의하면 C언어 한 문장(statement)은 대략 5-10줄 정도의 어셈블리 코드에 해당하는 작업을 수행하는 것과 비교하여, Tcl 스크립트 언어의 한 문장은 100-1000줄 정도의 어셈블리 코드에 해당되는 작업을 수행한다.

(풍부한 개발 라이브러리) 스크립트 언어들은 풍부한 개발 라이브러리들을 제공하고 있다. 이러한 개발 라이브러리들은 학교에서 개발된 이론적으로 무장된 훌륭한 언어들보다도 스크립트 언어를 선택하게 하는 또 다른 이유가 되고 있다. 다양한 개발 라이브러리를 통하여 기존의 많은 오픈 소스(open source) 소프트웨어 및 잘 알려진 라이브러리에 쉽게 접근, 개발이 가능하다.

(다이나믹 타이핑 dynamic typing) Lisp 또는 Prolog와 같이 스크립트 언어들은 다이나믹 타입 시스템을 제공한다. 실행 시간에 자료의 타입이 결정되며, 특정 타입의 변수는 실행 시간에 다른 타입의 변수로 사용될 수도 있다. 이는 컴파일시 최적화의 가능성을 떨어뜨리거나 타입 오류를 유발할 수도 있지만, 간결한 코드, 개발 시간 단축, 코드 통합 및 재사용성 증가 등의 장점을 제공하기도 한다.

(가비지 콜렉션 garbage collection) 스크립트 언어들은 메모리를 자동으로 관리해준다. 프로그램상에서 더 이상 참조하지 않는 자료들은 자동으로 수거되어 재사용이 가능하게 된다. 이를 통해서 시스템의 중요한 자원인 메모리의 관리에 대한 개발자의 부

표 1 각 줄은 C또는 Java와 Tcl로 각각으로 두 번 구현된 응용 프로그램을 나타낸다. Code Ratio는 각 언어로 구현된 라인수의 상대적인 값으로 >1인 값은 Tcl 구현 보다 C또는 Java로 구현된 소스 코드가 더 많은 라인 수로 구현되었음을 나타낸다. Effort Ratio는 상대적 개발 시간을 나타낸다. 각 구현은 다른 개발자들에 의해서 구현되었다[1].

Application (Contributor)	Comparison	Code Ratio	Effort Ratio	Comments
Database application (Ken Corey)	C++ version: 2 months Tcl version: 1 day		60	C++ version implemented first; Tcl version had more functionality.
Computer system test and installation (Andy Belsey)	C test application: 272 Klines, 120 months. C FIS application: 90 Klines, 60 months. Tcl/Perl version: 7.7K lines, 8 months	47	22	C version implemented first. Tcl/Perl version replaced both C applications.
Database library (Ken Corey)	C++ version: 2-3 months Tcl version: 1 week		8-12	C++ version implemented first.
Security scanner (Jim Graham)	C version: 3000 lines Tcl version: 300 lines	10		C version implemented first. Tcl version had more functionality.
Display oil well production curves (Dan Schenck)	C version: 3 months Tcl version: 2 weeks		6	Tcl version implemented first.
Query dispatcher (Paul Healy)	C version: 1200 lines, 4-8 weeks Tcl version: 500 lines, 1 week	2.5	4-8	C version implemented first, uncommented. Tcl version had comments, more functionality.
Spreadsheet tool	C version: 1460 lines Tcl version: 380 lines	4		Tcl version implemented first.
Simulator and GUI (Randy Wang)	Java version: 3400 lines, 3-4 weeks. Tcl version: 1600 lines, < 1 week.	2	3-4	Tcl version had 10-20% more functionality, was implemented first.

담을 덜어주고 간결한 코드 작성을 가능하게 해준다.

(명시적인 컴파일 과정의 제거) Python은 자바 언어와 유사하게 가상 기계상에서 수행되도록 설계되어 있다. Python 원시 코드는 실행 시간에 자동 컴파일 과정을 거쳐서 바이트 코드로 변환이 된 후, 가상 기계상에서 수행된다. 그러나, 자바와는 달리 사용자에 의한 명시적인 컴파일 과정은 없다. PERL 역시 원시 코드는 내부 자료 구조로 변환이 된 이후에 이 자료 구조를 통해서 수행이 되고 있지만, 명시적인 컴파일 과정은 없다. Tcl의 경우는 인터프리터 방식으로 컴파일 과정은 불필요하다. 이처럼 명시적 컴파일 과정이 생략됨으로써 기존의 코드 수정, 컴파일, 디버그의 반복으로 개발되는 소프트웨어 제조 과정은 단순화될 수 있다.

(상호 대화식 수행 환경) 보통, 탑레벨(Toplevel)이라고 불리는 상호 대화식 수행 환경을 제공함으로써 사용자는 개발 시 쉽게 함수를 구현, 테스트 할 수 있다. 사용자는 자신이 구현한 함수를 테스트해 볼 수 있으며, 확실하지 않은 시스템 라이브러리를 실험해 볼 수도 있다. 이를 통해서 개발 과정을 단순화 시키며 개발 시간을 단축 시킬 수 있다.

(언어 확장) 다른 프로그래밍 언어로 작성된 모듈을 호출할 수 있는 방법을 제공함으로써, 언어 확장 가능성을 제공하고 있다. 언어 차원에서 혹은 라이브러리 차원에서 부족한 기능들은 다른 언어로 작성된 모듈을 호출함으로써 언어 확장이 가능하다. 이 기능은 또한 특정 모듈의 성능 향상을 위해서 사용될 수도 있다. 기존의 비효율적인 모듈을 C/C++과 같은 시스템 프로그래밍 언어를 이용한 새로운 모듈로 교체할 수 있다. 이러한 언어 확장 과정은 자동화 툴을 통하여 수월하게 진행될 수 있다[2].

(내장형 언어) 다른 프로그래밍 언어 안에 기존의 스크립트 언어를 내장 시킴으로써 스크립트의 기능을 사용할 수 있다. 스크립트 언어를 확장할 것이냐 다른 시스템에 내장시킬 것인가는 개발자의 판단에 따라 달라지며, 언어 확장에서와 같이 모듈 구현을 다른 언어가 책임짐으로써 다른 언어 사이의 장점을 모두 이용할 수 있다.

4. 스크립트 언어의 응용 방안

아직 많은 개발자들은 스크립트 언어를 그들의 프로젝트에 채택하기를 주저하고 있다. 그러나, 이러한 결정은 스크립트 언어가 충분한 성능을 나타내지 못한다거나, 스크립트 언어를 통한 개발 생산성 향상에 대한 의구심에서 기인하는 경우가 많다.

스크립트 언어를 통한 개발 생산성의 한 예가 표 1에 나타나 있다. 상대적인 구현 코드의 라인 수 및 상대적인 개발 시간은 스크립트 언어가 개발 생산성에 많은 영향을 주고 있음을 나타낸다. 특히 개발 소스 코드의 감소는 소프트웨어 유지, 보수 및 검증에 유리할 수 있다. 스크립트로 구현된 특정 모듈이 빈번하게 사용되는 경우, 이 부분을 C/C++과 같은 시스템 프로그래밍 언어로 다시 작성함으로써 전체 소프트웨어 성능을 향상 시킬 수 있다. 앞에서 언급한 언어 확장 및 내장형 언어 형태를 사용하여 모듈 통합이 가능하다.

가장 성공적으로 스크립트 언어 및 통합 개발 방법을 채택한 곳은 웹 개발 스크립트 언어이다. PERL 스크립트 언어는 웹의 발전과 함께 사용자들의 많은 지지를 얻기 시작했다. 기존에 C로 작성되던 CGI 개발 과정은 PERL을 이용함으로써 매우 수월해졌다. 이러한 영향은 후에 전용 웹 개발 스크립트 언어인 PHP에 영향을 주었는데, PHP 또는 마이크로소프트의 ASP 등은 웹 개발 스크립트의 대표적인 예이다. 오늘날 웹 시스템을 성능상의 이유로 C로 구현하자고 주장하는 사람들은 찾아보기 힘들다. 특정 모듈 또는 컴포넌트는 C로 작성하지만 전체 컴포넌트 구성 조합 및 시스템 구성은 스크립트 언어가 담당하고 있다.

스크립트 언어는 또한 모듈 테스팅 도구로서 사용될 수 있다. C/C++과 같은 시스템 프로그래밍 언어로 작성된 모듈을 스크립트 언어에 통합한 후, 스크립트 언어로 테스팅 코드를 작성함으로써 기존의 모듈에 대한 테스팅 수행이 가능하다. 스크립트 언어의 상호 대화식 수행 환경을 통하여 기존의 디버거보다 조직화되고 자동화된 테스트가 가능하다.

5. 결 론

스크립트 언어가 개발 생산성에 미치는 효과는 간과할 수 없다. 스크립트 언어와 기존의 C/C++과 같은 시스템 프로그래밍 언어를 통합함으로써 생산성과 효율성 사이의 조율이 가능하다. 또한 스크립트 언어를 사용한 효과적인 모듈 테스팅도 가능하다.

최근의 웹 서버 시스템들은 스크립트 언어의 나아

갈 방향을 잘 보여주고 있다. 많은 웹 서버 시스템들은 시스템 프로그래밍 언어로 작성된 모듈(가령, 데이터베이스 등)과 스크립트로 작성된 부분이 적절히 조화를 이루고 있으며, 이런 연계 개발은 앞으로 더욱 활발하게 이루어질 것으로 예상된다.

참고문헌

[1] John K. Ousterhout, Scripting: Higher Level Programming for the 21st Century, IEEE Computer magazine, March, 1998.
[2] http://www.swig.org, SWIG(Simplified Wrapper and Interface Generator)
[3] J. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesley
[4] http://www.python.org, Python home page
[5] L. Wall, T. Christiansen, and R. Schwartz, Programming Perl, Second Edition, O'Reilly a
[6] http://www.perl.com, PERL home page

김 익 순

1994 포항공과대학교 전자계산학과 졸업
1995 한국과학기술원 전산학과 석사
2002 한국과학기술원 박사과정 졸업
현재 이화여자대학교 컴퓨터학과 연구전임 강사
관심분야 : dynamic code generation, automatic backend code generator, programming semantics, 및 automata theory
E-mail : iskim00@ewha.ac.kr

● 제30회 임시총회 및 춘계학술발표회 ●

- 일 자 : 2003년 4월 25~26일
- 장 소 : 제주대학교
- 논문모집 및 발표일정
  1) 논문접수 : 2003년 2월 10~28일
  2) 심사결과 통보 : 2003년 3월 18일
  3) 수정논문 접수마감 : 2003년 3월 25일
  4) 사전등록 : 2003년 3월 25일~4월 21일
  5) 논문발표 : 2003년 4월 25일, 26일
- 문 의 처 : 한국정보과학회 사무국
  Tel. 02-588-9246/7
  http://www.kiss.or.kr, E-mail:kiss@kiss.or.kr