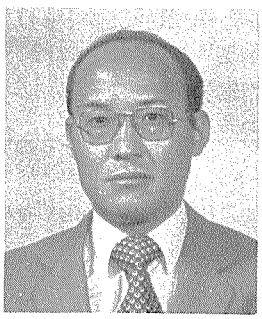

吳 吉 祿
韓國電子技術硏究所
컴퓨터연구부장 / 工博

Object-oriented 개념과 Smalltalk Virtual Machine 개발

"Smalltalk의 여러가지 특성을 잘 응용함으로 해서 Graphics, Image Editor, Object-Oriented Network 등의 개발에 효과적으로 이용, 가능하다."

1. 서 론

컴퓨터 시스템이 출현한 이래로 많은 프로그래밍언어들이 만들어졌다. (그림 1) 현재 컴퓨터 언어를 컴퓨터 구조로 볼때 다음과 같이 크게5가지로 분류한다.

1) Control Flow Architecture : 우리가 이제까지 사용하여온 대부분의 일반적인 컴퓨터언어를 대변하는 Architecture로 제5세대 컴퓨터에서는 Evolutionary Approach로 본다.

2) Data Flow Architecture : ID, LUCID, VAL, VALID와 같은 Single-assignment 언어로서 Input Operand가 모두 사용가능하게 되면 즉시 명령이 수행되도록 한 병렬처리 조직이다.

3) Reduction Architecture : Pure LISP, SASL, FD와 같은 Applicative 언어로서 어떤 결과의 요구만으로 그에 필요한 값을 생성하게 될 Instruction을 수행시킨다. 여기에는 크게 String Reduction과 Graph Reduction의 2가지 방법이 있다.

4) Actor Architecture : Smalltalk와 같은 Object-oriented 언어가 대표적인 예로서 Message가 도착하면 Instrction이 수행된다.

5) Logic a Hitecture : Prolog 언어와 같이 Predicate Logic을 이용하여 Object와 Object들 사이의 Relation을 가지는 문제를 해결하는데 사용되는 언어로 제5세대 컴퓨터에서 Revolutionary Approach로 본다.

그리고 이러한 프로그래밍 언어들을 인공지능의 측면과 연결시켜 볼때 LISP, PROLOG, Smalltalk와 같은 Programming 언어를 AI 언어로 부른다.

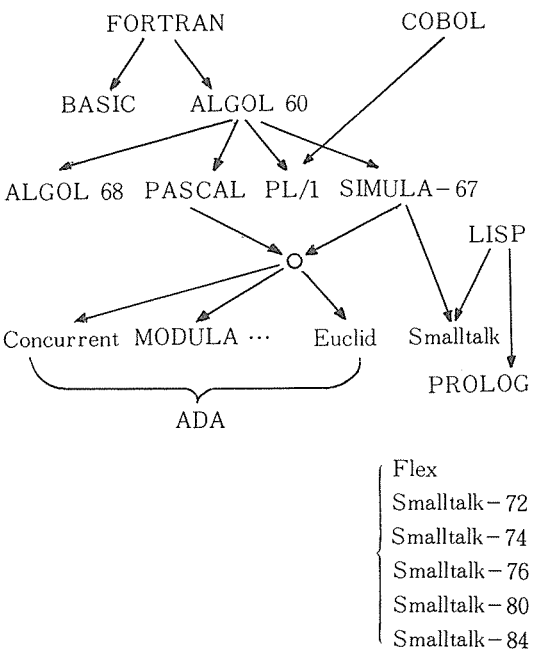

그림 1 Language Flow Diagram

2. Object-Oriented Language

2.1 Object-Oriented 개념의 출현

Type이나 Object와 같은 Object-Oriented의 개념을 사용한 것은 지금으로부터 약 70년전 B. Russell과 A. Whitehead에 의하여 연구된 Type Theory에서였다. 그들은 이 이론이 수학의 논리적인 기초하에서 존재하는 패러독스와 모호함을 해결하는데 큰 도움이 된다는 것을 깨달았다. 이러한 Type Theory는 오늘날 High Order Predicate Logic의 형태에서 잘 이용되고 있으며 프로그래밍 언어에서 발견되는 Object-Oriented의 개념에 많은 기초가 되었다.

수학자와 이론적인 컴퓨터 과학자들 사이에 인기있는 또하나의 이론은 40년전에 개발된 Category Theory인데 수학적인 Object와 Morphism (Object에서 명확히 정의될 수 있는 Operation들)의 집합으로 구성되어 있다. Category Theory는 다음과 같은 Object-Oriented 특성을 지닌다.

1) Abstract Class 사이의 Relation, Allowable Operation, Object가 정확히 정의된다.

2) 한 Category에서 다른 Category로 변환하기 위한 Mechanism을 제공한다.

즉, 이 이론의 Programming 에서의 유사성은 주어진 Class의 한 Object에서 다른 Class의 Object로 명확히 변화시킬 수 있는 것과 같다.

실제로 ALGOL 60의 Type, Simula의 Object와 Class, ADA의 Package, Task과 Generic, Smalltalk의 Object, Message 등은 이러한 Object Oriented의 개념을 포함하는 것이다. 그리하여 ALGOL 60과 Type의 이론에 힘입어 SIMULA와 Smalltalk와 같은 Object-Oriented 언어가 나타났다.

2.2 Procedure-Oriented와 Object-Oriented

전통적인 의미에서 소프트웨어 시스템이라는 것은 어떤 정보를 나타내는 Data와 Data를 조작하기 위한 Procedure의 집합을 말한다. 즉 다시말하면 Data란 소프트웨어에 의해 조작되는 정보를 말하고 Procedure란 Software의 단위를 말하는데 Data/Procedure-Oriented Software의 경우 이러한 Data와 Procedure가 분리되어 있다. 이에 반하여, Object-Oriented System은 Object라는 단일 형태의 Entity를 가지는데 여기서 Object란 Object 의 조작을 위한 방법과 정보를 포함하는 Package로서 한 Object내에는 Data와 Procedure가 공존한다. 즉, Procedure-Oriented와 Object-Oriented의 차이를 간단히 얘기하면 Procedure-Oriented에서는 Data와 Procedure가 분

리되어 존재하고,

Object-Oriented의 경우는 Object내에 data와 Procedure가 동시에 존재한다.

2.3 Evolutionary와 Revolutionary

제5세대 컴퓨터에서의 분류에서처럼 Object-Oriented Programming도 그 언어내에 포함하는 Programming Technique의 정도에 따라 Revolutionary와 Evolutionary로 분류할 수 있다. Smalltalk와 같은 순수한 Object-Oriented Language는 Revolutionary Approach를 나타내고 개념적으로는 간단하다는 이점이 있다.

그리고 일상으로 사용하는 보통의 언어에다 Object-Oriented 개념을 추가한 Evolutionary Approach가 있다. 이것을 Hybrid Language라고하며 Objective-C, OOPC, Flavors, Clascal 같은 언어들이 있다. 이러한 언어는 개념적으로 Smalltalk와 동일함을 제공하지는 않지만 상당한 이점을 가지고 있다. 이러한 언어는 Smalltalk와 같은 순수한 Object-Oriented 언어에서 받아들일 수 없는 Production Programming을 위하여 사용될 수 있다. Smalltalk는 다양하게 Object에 치중했기 때문에 보통의 Operator와 Operand를 제거하여, 일상의 Programming Technique을 필요로 할때 어려움이 없지 않다. Hybrid Language는 Object-Oriented Language에서의 강력한 Tool을 추가한 관계로 상당히 편리하고 효율적으로 종전의 여러가지Tool이나 Program, Database 등을 쉽게 쓸 수 있다.

2.4 주요한 용어

Object-Oriented Software에서 사용되는 주요한 용어를 간단히 살펴보면 다음과 같다.

1) Object : Software System의 구성요소 즉, Number, Character String들, Queue, Dictionary, Rectangle, File Directory, Text Editor등을 나타내는 것으로 한 Oject는 어떤 Private Memory와 Operation의 집합으로 구성되어 있다. 즉 다시 말하면 Information과 그것의 조직을 표현하는 Package

2) Message : Object에 대한 Operation의 수행을 위한 요청으로 Message는 무엇을 할 것인가를 묘사한다.

3) method : Processor에 의해 수행되어야 할 action의 연속을 묘사한다.

4) class : 하나 이상의 비슷한 Object들을 묘사한다.

5) instance : 특별한 Class에 의해 표현되는 Object

6) Inheritance : Superclass에 대한 모든 것을 이어받는 Subclass

3. Smalltalk Language

3.1 Smalltalk의 출현

Smalltalk는 70년대초에 Xerox PARC의 Alan Kay에 의해 개발되었다. Smalltalk 언어에서의 Object나 Message, Class 등의 기본적인 개념은 SIMULA에서 부터 왔다. SIMULA 언어는 사용자들에게 Object-Oriented System을 만들 수 있도록 허용하지만, Number, Boolean, 기본적인 Data Structure나 Control Structure 등은 Data/Procedure-Oriented인 ALGOL 언어의 기본적인 개념을 사용한다.

Flex System, Smalltalk-72, Smalltalk-74, Smalltalk-76등은 프로그램 환경하에서의 Element에 Object-Oriented의 개념을 확장해 왔다. 예를 들면 Smalltalk-72에서는 arithmetic, List Structure, Control Structure 등이 Object나 message로서 표현되었고 Smalltalk-74에서는 Object로서의 Class가 포함되었다. Smalltalk-76에서는 Class 사이의 관계를 표현할 수 있게 했다. 현재까지, Smalltalk를 개발

한 몇가지 예를 알아보면 다음과 같다.

1) Tektronix : 8MHz의 MC68000 machine에서 Pascal과 기계어를 사용하여 설치

2) Hewlett-Packward : 4.1BSD UNIX와 4 Mbyte의 주기억용량을 가진 VAX 11/780 에서 C언어를 사용하여 설치

3) Xerox : Xerox Dorado System에서 microcode와 기계어를 사용하여 설치

4) U. C. Berkeley : 4.1BSD UNIX의 VAX 11/780에서 C 언어를 이용하여 설치

5) Tokyo University : 4.1BSD의 VAX 11/780과 Sun Workstation을 Ethernet로 연결하고 C언어를 이용하여 설치

3.2 Smalltalk Virtual Machine

Smalltalk 시스템이 출현한 가장 큰 이유중의 하나는 User-Friendly System을 만들어 보자는데 있었다. Smalltalk System은 대형의 응용프로그램을 개발하기에 좋은 강력한 시스템으로서 Compiler, Debugger, Storage Management System, Text와 Picture Editor, File System 등을 포함한다. 그리하여 Smalltalk 시스템은 이러한 기능을 효율적으로 수행하기 위하여 Bit-Mappped Graphics를 채택하며, 상호대화적인 User Interface, 사용자의 Program에 대한 융통성을 증가시키기 위해 window나 menu, Scrollbar, mouse 등의 기능을 채택하고 있다. Smalltalk System은 크게 Virtual machine과 Virtual Image로 나눌 수 있다. Virtual Machine이란 특정한 Hardware System을 위한 가상적인 machine으로서 Smalltalk Virtual Machine은 Interface Storage Manager, I/O Device를 위한 Primitive Subroutine으로 구성되어 있다. Virtual Image는 기본적인 자료 구조나 기본적인 Graphics와 Text, Compiler, Decompiler, Debugger와 User Interface를 위한 Class들을 포함하는 Object들의 집합이다.

3.3 Smalltalk-80 System

Smalltalk System은 서로간에 Message를 주고 받는 Object로 구성되어 있으므로 보내져야 할 Message나 Message를 받았을때 일어나는 일을 묘사함으로써 System을 구성할 수 있다. 여기서 Message란 Smalltalk-80 언어의 문법에 맞는 문자들의 연속으로 이루어지는 표현이다. Message-Sending의 표현은 Message의 Receiver, Selector, Argument를 표현한다. 어떤 표현이 수행될 그 표현을 묘사하는 message는 그것의 Recover에 전달된다. Message는 다음과 같은 종류가 있다.

1) Unary Message : Argument를 가지지 않는 Message, Unary Message는 Unary Selector라고 부르는 Single Indentifier로 구성되어 있다. (예 1)

2) Binary Message : (+, −, *, /) 와 같은 Binary Selector와 하나의 Argument를 가지는 Message (예 2)

3) Keyword message : 하나 이상의 Argument와 Colon으로 끝나는 Keyword인 Selector로 구성된 Message (예 3, 예 4)

Message가 보내질때, Receiver의 Class에 의해 결정되는 Method를 요구한다. 이 요구된 Method는 항상 결과 (하나의 Object)를 돌려ssage를 받았을 때 일어나는 것을 묘사하는 Method의 집합을 제공한다. Method는 다음과 같이 세부분으로 나누어져 있다.

*Massage Pattern

*어떤 일시적인 변수의 이름

*어떤 Expression

여기서 세부분의 Method는 Vertical Bar(-)에 의해 분리된다. Message Pattern은 Selector와 Argument를 위한 이름으로 구성되어 있다. 각 표현들은 Period (.)에 의해 분리되고

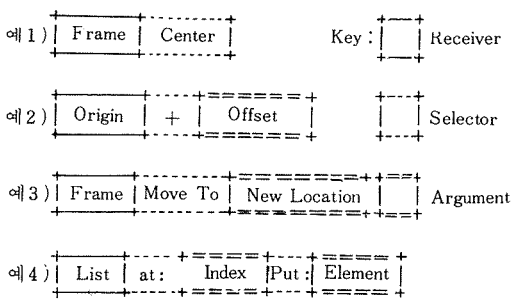

그림2. Message의 예

준다. 또 Message의 결과는 다른 Message 를 위한 Receiver나. Argument로 사용될 수 있다. 또 기본적인 Parsing Rule은 Unary Message와 Binary Message는 Left에서 Right로 수행된다. 그러나 Keyword Message의 경우는 Left에서 Right로 Parsing 하는 Rule이 없다. 즉, 만약에 한 Keyword Message가 다른 Keyword Message의 Receiver나 Argument로 사용되면, 괄호로 묶여져야 한다. 또한 표현안에 Unary, Binary, Keyword Message가 동시에 있을때 Unary Message, Binary Message, Keyword Message순으로 보내진다.

Window Frame Center

와 같은 예에서 첫번째 Message는 Unary Selector 「Frame」이 「Window」라는 Object로 보내지고, Unary Message 「Center」는 Window Frame의 결과에 보내진다.

Index+Offset* 2

여기서 Binary Message 「+Offset」이 Index라는 Object에 보내진 결과가 Binary Message 「* 2 」를 위한 Receiver가 된다.

Index+ (Offset * 2)

와 같은 경우는 Offset에 보내진 Binary Message 「* 2 」의 결과가 Selector 「+」와 Index를 Receiver로 가지는 Binary Message의 Argument로 사용된다.

Frame Center+Window Offset-Index

와 같은 예의 경우는 Unary Message와 Binary Message가 중복된 경우인데

((Frame Center)+(Window Offset))-Index

로 표현하면 쉽게 알 수 있다. 즉 다시말하면 Frame에 보내진 Binary Message Center 의 결과가 Selector가 +이고 Argument는 Unary Message Offset이 Window에 보내짐으로써 얻어진 결과인 Binary Message의 Receiver가 된다. 또 이 전체의 결과는 「-Index」의 Receiver가 된다.

Big Frame Height : Small Frame Height * 2

Big Frame Height : (SmallFrame Height) * 2

이 경우에도 마찬가지 방법으로 수행되며 두예의 수행결과는 동일하다.

Class란 Instance로 불리는 Object의 집합을 묘사하고, 각 Instance는 Instance Variable의 집합을 가진다. Class는 또한 Instance가 Me

마지막 표현은 Up Arrow(↑)에 의해 앞선다. 그림3의 예에서 Selector +를 위한 Method에서 Message Pattern은 +apoint 이고, temporary Variable Name은 sumX, sumY, 그리고 3개의 표현중 마지막 것은↑에 의해 앞서고 있다. Message가 Instance에 의해 받아들

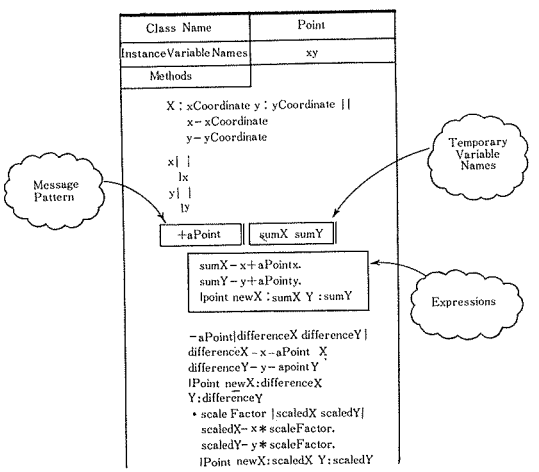

그림3. class Point에 대한 class template

여졌을 때, 같은 Selector를 포함하는 Message Pattern을 포함하는 Method가 수행된다. 예를 들면,

Offset+Frame Center

에서 Offset 이 표현에서 Point의 Instance 라고 가정하면, Message Pattern이+apoint인 Method가 수행된다. Argument를 가지는 Selector에 대해서 Message Pattern은 또한 Message에서 나타내는 Argument의 어디든지 Argument Name을 포함한다. Method가 Message에 의해 수행될때, Method에서 Argument Name은 그 Message의 실제 Argument 에 관계된다. 위 예에서 aPoint 는 Frame Center의 결과가 될 것이다.

이러한 형태의 Message를 가지는 Smalltalk Virtual Machine과 그와 일치되는 Bytecode의 집합은 Stack-oriented Operation을 한다. 그러므로 Object Pointer들은 Stack에 Push 되거나 Pop된다. Message가 보내지면 Stack의 Top에 있는 몇개의 Element가 그 Method의 Receiver나 Argument로서 사용된다. Stack에서의 수행방법을 예를 들면 다음과 같다.

예1)

Expression : 3 * (4 + 5)

Bytecode Stack Contents After Execution

〔1〕 Push 3 (3)

〔2〕 Push 4 (3 4)

〔3〕 Push 5 (3 4 5)---〉 Top of Stack

〔4〕 Send+ (3 9)

〔5〕 Send * (27)

예 2)

Expression: a > 4 if True: 〔a←a−1〕

Bytecode

〔1〕 Push 4

〔2〕 Push a

〔3〕 Send >

〔4〕 Jump to 〔10〕 if the top of the stack is false.

〔5〕 Push a

〔6〕 Push 1

〔7〕 Send−

〔8〕 Store into a

〔9〕 Pop

〔10〕 〈the Next Bytecode〉

4. 결 론

Smalltalk 언어에 대한 설명을 짧은 지면하에서 이해되게끔 하는 것은 어려운 일이다. 그

래서 여기서는 Smalltalk의 역사적 배경과 기본적인 개념만을 소개하는데 그치고 있다. 그만큼 Smalltalk 언어가 이제 까지의 다른 언어와는 그 기본 개념이 많이 다르고, 이때문에 이제까지 일상의 컴퓨터 언어를 써오던 사람들에게는 오히려 더 혼동을 일으키기도 한다.

Smalltalk의 여러가지 특징을 잘 응용함으로써 Graphics나 Image Editor, 그리고 Object-Oriented Network 등의 개발에 효과적으로 이용할 수 있다. 그러나, Smalltalk를 효율적으로 설치하기 위해서 대량의 주기억 용량과 빠른 Machine Speed가 요구되고 있기 때문에 아직 이렇다할 상업적인 제품은 나와있지 않으나 그 기본적인 Idea는 Macintosh나 Lisa 등 여러 시스템에서 이용해 왔다. 앞으로 Smalltalk는 그 자체가 하나의 Language의 역할을 하며 Smalltalk System이 또 하나의 Operating System의 역할을 함으로 인하여 Workstation 같은 Machine 위에 설치되어서 Smalltalk Machine을 구성하는 방향으로 나아갈 것이며 이를 응용하여 Object-Oriented Network의 실현을 이루는 방향으로 나아갈 것이다.

앞으로 전자기술 연구소는 인공지능에 대한 연구·개발 환경을 만들기 위하여 국가 특정연구 사업으로 매년 2억5천만원 규모로 3개년 동안에 이런 Machine을 개발할 계획이다. 첫째 연도에는 UNIX하에서 Smalltalk Language Porting, Multiwindow Text Editor개발, Smalltalk 응용시스템개발등이고; 2, 3次 연도에는 Bare Machine에서의 Smalltalk Porting, Bit Slice CPU Board Design Prototype구성 등 H/W제작이중심이되고 그후에 System Tuning 및 Test와 Graphics 응용시스템 개발 등 양산화 시스템 개발에 중점을 두게 될 것이다.

References

〔1〕 P. C. Treleaven, 「The New Generation of Computer Architecture」 ACM, 1983.

〔2〕 R. S. Freedman, 「The Common Sense of Object Oriented Language」, Computer Design, February 1983.

〔3〕 「The Smalltalk-80 System」, BYTE, August 1981.

〔4〕 D. Robson, 「Object-Oriented Software System」, BYTE, August 1981.

〔5〕 A. Goldberg and D. Robson. 「Smalltalk -80 :The Language and Its Implementation」, Addison-Wesley Pub., 1983.

〔6〕 G. Krasner, 「Smalltalk- 80; Bits of History, Words of Advice」, Addison-Wesley Pub., 1983.