5Q-5

CLOS準拠のHiOBJ-2の概要

湯浦克彦((株)日立製作所 中央研究所)
○堺原健((株)日立製作所ソフトウエア工場)
高橋久(日立超LSIエンジニアリング(株))
船津隆(日立ソフトウエアエンジニアリング(株))

1．はじめに

オブジェクト指向は、ユーザーインターフェース構築に盛んに利用されているほか、システム記述、データベース等の分野で次世代の記述言語としての期待が大きい。

Lisp言語においては、Flavorsをはじめとしたシステムでオブジェクト指向機能を早くから取り入れている。これらのシステムでは、多重継承、メソド結合などの拡張機能を提案していくなど、オブジェクト指向のトップランナーとしての役割を果たしてきた[1]。Common Lispによる標準化においてはオブジェクト指向の拡張も採り上げられ、CLOS(Common Lisp Object System)[2]により米国での標準化が図られる模様である。CLOSは、従来のLisp上での各提案の長所を吸収した、豊富かつ汎用性の高い言語である。CLOS言語およびその利用法の蓄積が、今後のオブジェクト指向の高度化、実用化をリードしていくと考えられる。　以上のような背景をもとに、報告者らは、CLOS仕様によるCommon Lispオブジェクト拡張システムHiOBJ-2の開発に着手した。母体となるCommon Lisp処理系は、報告者らが以前に開発した高速処理系HiLISP[3]である。

2．ＣＬＯＳの特徴と利用法

CLOSには下記の２つの特徴があると考えられる。

（１）多重継承、メソド結合を用いた差分プログラムが可能である。

メソド結合とは、同名のメソド群を組合せて動作させる機能である。CLOSではメソド結合のタイプとしてデーモン的メソドを付加できる標準タイプ、既定の各種メタ制御が可能な組込タイプのほか、ユーザに制御法を記述させる宣言タイプまで用意している。多重継承とメソド結合を合わせて用いることにより、既存のプログラムに手を加えず追加のみで機能を拡張していくことが容易である[4]。図１(1)の例の原形はノードとアークの編集プログラムで、ノードの移動などが可能である。ここに、移動の前（:before）に関係ノードに移動を伝播するメソドを追加することで、関係ノードも含んだ連鎖移動を可能にしている。

（２）クラス変更、メソド追加などを随時行う高度対話プログラミングが可能である。

クラス変更とは、既に定義されたクラスを再定義して継承木の変更やスロットの追加を行う機能である。このほか、メソド追加削除等、CLOSでは動的にプログラムを変更する機能が豊富である。図１(2)の例では、図１(1)と同じ原形より、ノードの属性を表示ラベルの属性とその他の属性に分けて構成し直している。そのうえで、ラベル以外の属性を再利用して、パタン付きノードへの拡張を行っている。

またクラス変更においては、旧クラス定義によるインスタンスを自動的に修正する機能も含んでいる。図１(3)の例では、これも同じ原形より、クラス再定義により丸いノードのクラスnodeを四角いノードのクラスに変更している。ここでクラスnodeのインスタンスA,Bが丸より四角へと自動的に変更されるのであるが、元の丸の半径より四角の高さ、幅への変換法は、再定義に先だってメソドとして指定することができる。

3．ＨｉＯＢＪ－２の開発

3．1　ＣＬＯＳ実現上の課題

ＣＬＯＳの実現では以下の２点が問題となった。

（１）メソド結合をはじめとしての処理系の基本に関わる部分で汎用の機能が多いので、素直に作ると基本制御が重いものになってしまう。

（２）クラス変更等動的な変更機能のため、処理系全般にわたっていつでも円滑に制御を変更する枠組みを確保しなければならない。

3．2　ＨｉＯＢＪ－２

HiOBJ-2の構成を図２に示す。HiOBJ-2は以下の方針に従い設計した。

（１）高度対話プログラミング向きのシステムを目指し、まず動的機能を高速に実現することを重視する。

（２）差分プログラミングを十分活かすため、複雑なメソド結合を利用しても、性能が大きく落ちないようにする。

Object-Oriented Programming with HiOBJ-2 based on CLOS
Katsuhiko YUURA[1], Ken SAKAIBARA[2], Hisashi TAKAHASHI[3], Takashi FUNATSU[4]
[1]Central Research Laboratory, Hitachi, Ltd., [2]Software Works, Hitachi, Ltd.,
[3]Hitachi VLSI Engineering Corp., [4]Hitachi Software Engineering Co. Ltd.

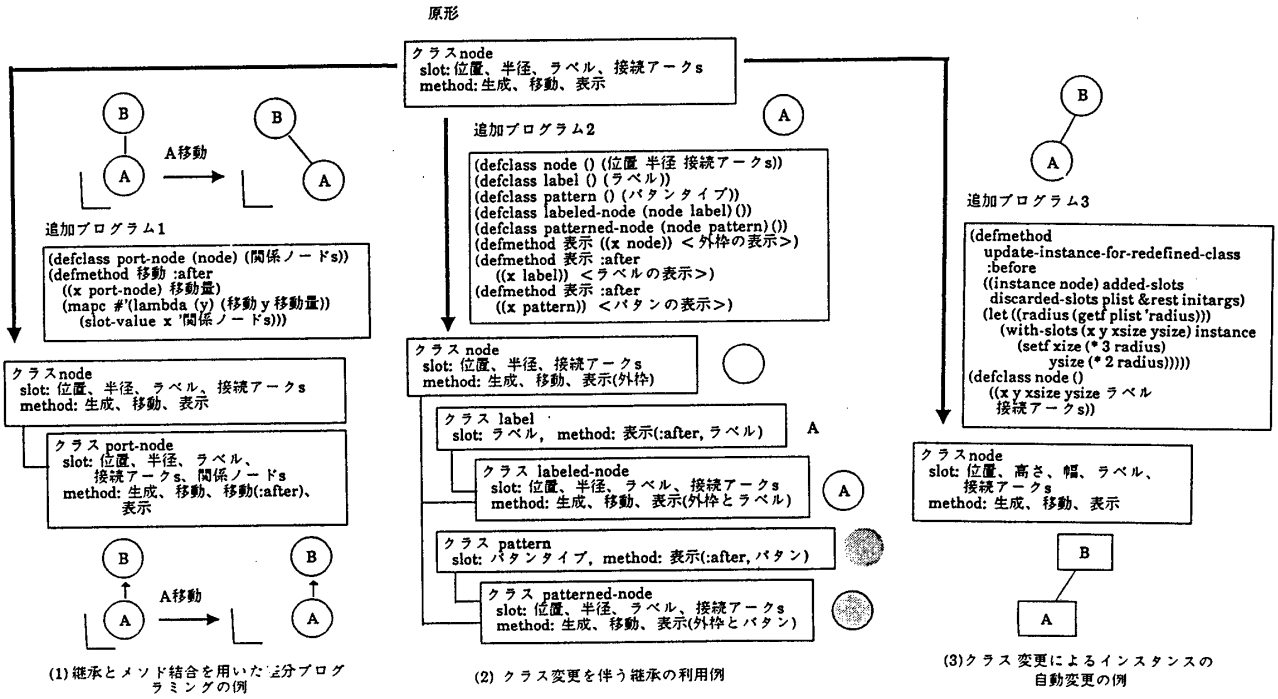

図1 多重継承、メソド結合、クラス変更の利用例

そこで、総称関数制御では、一度計算された実効メソド等の記録参照（キャッシング）において、クラス変更やメソド追加が起こってもキャッシュ表の変更を少なくて済む方式を開発した[6]。また、スロット参照では、クラス再定義での変更処理を最小限に抑えることにして、クラス定義データ（スロット位置、継承、再定義）を高速に参照できるようにクラス定義の世代管理方式を開発した[6]。

4．おわりに

CLOSの利用法と処理系実現の考え方について述べた。プロトタイプを作成し利用を開始している。動的変更機能は快適に使えるし、実行性能もLispで型毎の処理を分ける記述をするよりは明らかに高速である。

設計分野でのエキスパートシステム、マンマシン構築ツール等での利用を広めたいと考えている。

参考文献

1)梅村、外：Lisp上のオブジェクト指向プログラミング、情報処理、vol.29 no.4、1988.

2)D.G.Bobrow et al.;Common Lisp Object System Specification,Draft X3J13 Document 88-002R,1988.

3)湯浦、外：高速Common Lisp-HiLISPの実現、情報処理学会第33回全国大会、2E-1、1986.

4)湯浦、外：CLOSの記述性について、情報処理学会第39回全国大会、1989.

5)湯浦、外：HiOBJ-2における総称関数制御方式、情報処理学会第39回全国大会、1989.

6)高橋、外：HiOBJ-2におけるクラス変更方式、情報処理学会第39回全国大会、1989.

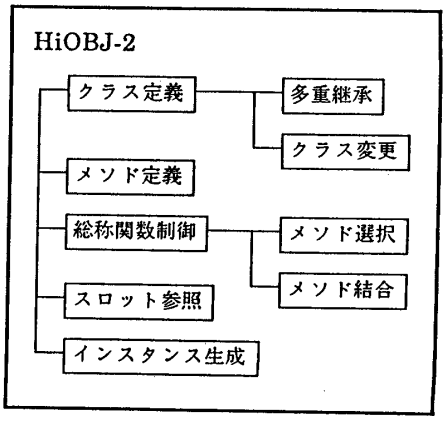

図2 HiOBJ-2の構成