5D-5

構造的な抽象化機能を持つオブジェクト指向言語 Ｏｎｄｉｎｅの概要

荻原剛志　梶原洋一　長野伸一　有澤　誠
山梨大学工学部　計算機科学科

１．はじめに

オブジェクト指向の考え方による利点は、大きく次の３つに分けられよう。

1. 人間の思考に近いモデルが構築できること
2. プログラムのモジュール化、差分プログラミングなどが実現できること
3. オブジェクトの考え方が、プログラムの並列性を記述するのにも適していること

しかし、従来の『オブジェクト指向型言語』が、これらの要求を満たすのに十分な機能を持っているかどうかは疑問である。

そこで本研究では、オブジェクト指向型と言われている言語の問題点を検討し、再構築することにより、より強力な言語システムの可能性を探ることとした。このシステムの名称を、Ｏｎｄｉｎｅ（オンディーヌ）とする。

Ｏｎｄｉｎｅの中心となる構想は以下の３つである。

1. オブジェクトの性質についての解釈
2. クラス階層と継承、型の実現方法
3. オブジェクトの並列実行方式

本稿ではこのうち、オブジェクト、クラス、型についての概略を述べる。

２．クラスとオブジェクトについて

Ｏｎｄｉｎｅではクラスはオブジェクトではなく、インスタンスであるオブジェクトの内容を定義した宣言としてとらえる。すべてのオブジェクトはクラス、あるいは他のオブジェクトのコピーとして生成し、必要に応じて修正を加える。

この方式には、クラス変数、クラスメソッド、あるいはメタクラスといった概念がない。オブジェクトはクラスのインスタンスのみであり、すっきりした構成とすることができる。

さらに、コンスタント（定数）、関数もクラスの特別なものとして考える。これら自体は単なる定義であり、メッセージを受け取ったりすることはできない。しかし、これらに対してメッセージを送ることは、そのコピーのオブジェクトにメッセージを送ることであると考えるのである。このコピーのオブジェクトは一時的に生成し、処理の終了とともに消滅する。これにより、構造的なコンスタント、副作用のない関数を実現できる。

コンスタント、関数については今まであまり明確な定義がなかったが、この考え方はこれらに対し、統一的な見解を与えるものであると考えられる。

Ｏｎｄｉｎｅの大きな特徴のひとつは、メッセージの引数の受渡しを、原則としてコピーで行うことである。また、オブジェクトは内部に、そのオブジェクト固有の情報を持つ構成オブジェクト（いわゆるインスタンス変数）を持つことができるが、この構成オブジェクトが他のオブジェクトを示すこともできないようになっている。あるオブジェクトが自分の外部のオブジェクトにメッセージを送る場合には、そのことをはっきりと明示しなければならないのである。

このような考え方を採用したのは、オブジェクトを渡した先での副作用、特に並列処理時の相互排除を考慮したためである。

３．クラス階層の記述方法

オブジェクト指向言語の差分プログラミングの能力は、継承を利用したクラス階層によるものである。しかし、問題点も多い。主なものとして、情報隠蔽やクラス管理が不十分であること、継承機構によって『ひとまとまりのデータに対する手続きはまとめて書く』という原則が損なわれやすいこと、がある。

これらの問題点を解決するために、まずクラス階層を、継承の意味を表現する構造的な方法で定義できるようにするべきであると考えた。

スーパークラスとサブクラス間の関係を端的に表現するには、木構造をとるのが自然である（図１）。しかし、多重継承を認めるとすると木構造では対応できない。

多重継承の場合には、スーパークラスの機能を拡充するというよりも、既存のクラスから新たなクラ

An Object Oriented Language Ondine and it's Structured Abstraction
Takeshi OGIHARA, Yoichi KAJIHARA, Shin-ichi NAGANO, Makoto ARISAWA
YAMANASHI Univ.

スを作成するといった意味あいが強いと考えられる。そこで、多重継承の場合も含め、スーパークラスの具体化と見るのはふさわしくないと考えられるものは外部へ出すこともできることにする（図２）。

```
class A ...{define}... endclass A
  subclass
    class B ... endclass B
      subclass
        class B1 ... endclass B1;
        class B2 ... endclass B2;
      end B;
    class C ... endclass C;
  end A;
```

図１　クラスの木構造

```
class X ... endclass X
  subclass
    class X1 ... endclass X1;
  end X;

class Y
  inherit B2, X1;
  ...
endclass Y;
```

図２　多重継承の場合

この方法は、サブクラスをスーパークラスの具体化が進んだものとする見方と、スーパークラスをサブクラスを構成するための部品とする見方の統合となっているともいえよう。

Ｏｎｄｉｎｅではオブジェクトのインタフェース（仕様）と本体の定義を別々に書くことができる。クラス階層の宣言にはインタフェースのみを記述してもよい。

従来のオブジェクト指向言語では、サブクラスのメソッドからスーパークラスやメッセージの引数の内部情報にアクセスすることができ、情報隠蔽が完全ではなかった。この点については、原則としてインタフェースしか見せないというように、厳しくすることとした。ただし、スーパークラスの内容は、継承の記述の際にその旨を宣言すればサブクラスで参照できる。

クラス名は、通常はグローバルで、オブジェクトが生成可能であるが、サブクラスを構成するだけのための中間的なクラスなどは、その部分木内でローカルとしたり、オブジェクトが生成できないように指定したりすることもできる。

４．　型の導入

Ｏｎｄｉｎｅでは、翻訳時の誤り検出、概念整理の観点から、オブジェクトに型を導入した。型はクラスの階層構造を利用して定義する。ここで、クラス定義と別に、型定義という仕組みを考え、次のことを規則とする。

1. クラスはひとつの型である。
2. ある型から別の型を作ることができる。これは継承による定義の一種である。しかし、新しい構成変数を付け加えること、多重継承はできない。従って、２つの型は構造的にまったく同じものである。
3. ある型から型定義で作った型は、もとの型と互換である。ここで、「互換」とは、メッセージの引数として相互にオブジェクトのコピーが行えることである。
4. ある型から作った複数の型の間には互換性がない。つまり、「親子」間では互換性があるが、「子供」間では互換性がないのである。

以上述べた方法によって、型の導入目的の基本的な要求を満たすことができる。即ち、

1. 異なる構造のオブジェクトを違う型として、相互の演算を禁止することができる。
2. 同じデータ構造を持ちながら、異なる意味を持ったオブジェクトを、違う型として、相互の演算を禁止することができる。
3. 2. の場合において、意味は異なるが、規制を緩めた方が実用的であるものについて、ある程度の融通性を持たせることができる。

この型の方式は、先に述べたクラス、コンスタント、関数の概念とあわせて、統一的な構造としてとらえることができる。

５．　おわりに

Ｏｎｄｉｎｅの仕様はそれほど大きくはないものの、新しい考え方が多いため、すべてを簡潔に記述することができなかった。並列処理など、今回触れることができなかった点も含め、別の機会に述べたいと思う。

現在、Ｏｎｄｉｎｅは VAX-11/785 の UNIX*(4.2 BSD) 上で開発中である。

末筆であるが、多くの助言を頂いた井内稔技官に感謝する。

参考文献

[1] 荻原剛志：オブジェクト指向言語 Ondine の抽象化機構，ソフトウェア基礎論研究会，18-4，(1986. 9.30)

[2] 久野靖：プログラム言語Mistyの試作版処理系と使用経験，ＷＯＯＣ'86資料

＊ ＵＮＩＸはＡＴ＆Ｔベル研究所の商標である。