3Q-1

# 知能処理向きオブジェクト指向言語に必要な機能に関する一考察

塚田晴史　杉村利明
NTTヒューマンインタフェース研究所

## 1.はじめに

オブジェクト指向言語の使用目的には大きく２つの流れがある。１つはＯＳ，ネットワークなどシステム記述用、もう１つは知能処理用である。この２つではオブジェクトへの要求条件が異なる。前者のオブジェクトが実行開始時から構造が不変な静的なものであるのに対し、後者のオブジェクトは動的である。クラス構造，オブジェクトのクラスなどは、知識の変化に伴いプログラムの実行中にも変化しうる。

このうち知能処理に必要な機能を考察する。オブジェクトの動的変更機能の他に、もう１つ重要なのが並列機能である。知能処理の分野では、分散人工知能に見られるように、問題分野に並列性が含まれるケースが多い。こうした問題をプログラムで表現するためには、言語に並列機能が不可欠となる。

本稿では知能処理のためのオブジェクト指向言語に必要な機能として、オブジェクトの並列性とプログラム実行中の動的変更を述べ、実現の問題点として、２つの機能の干渉を議論する。

## 2.オブジェクトの並列機能

オブジェクト指向においては、オブジェクトを並列単位と考え、並列性はオブジェクト間通信で表現されるのが一般的である。通信機能としては、同期／非同期のメッセージ送信（一方向）が基本で、返答待ちの同期機構を加えて、メッセージ交信（双方向）が実現されている。

先にあげた３つの言語では Harmonia，ABCL/1 が並列機能を持つ。Harmonia ではメイルボックスを用いた基本機能のみを提供し、他の機能はそれらの組合せで実現される。また ABCL/1 では３種類のメッセージ伝達、過去型（非同期送信），現在型（同期送信），未来型（非同期交信）を基本としている。

また並列の単位に関しては、オブジェクトの中で同時に複数のメッセージの実行を行う多重アクティビティを許す考え方もある。多重アクティビティの導入により実行効率は上がるが、オブジェクト内での同期機構が必要になる欠点がある。Harmonia，ABCL/1 は単一アクティビティの考えに基づく。

---

A Study on Object Oriented Languages for Knowledge Processing

Seishi TSUKADA　Toshiaki SUGIMURA
NTT　Human Interface Laboratories

並列動作の単位を分かりやすいものにし、プログラミングを容易にするという観点から、並列知能処理のための言語システムは、単一アクティビティに基づく基本機能のみを提供し、あとはユーザがそれらを組み合わせて自分に必用な機能を実現する方法がよいと考えらえる。

## 3.オブジェクトの動的変更機能

知能処理において、知識，状況，推論などは確定したものではない。これらの変更は一般にオブジェクトの変化で表現される。例えば今まで三角形だと思っていたオブジェクトが、新しい推論から四角形だと判れば、そのオブジェクトのクラスを三角形から四角形に変える操作が必要になる。オブジェクトのクラス変更の他にクラスのスロットの追加，削除、スーパークラスの変更、メソッドの追加，削除などの機能が必要である。また変更を関係箇所に伝える機能も必要になる。

一方、別のオブジェクトの作り方もある。クラスを定義してからインスタンスオブジェクトを作る標準的な方法のほかに、具体例から一般型へ、即ち見本となるオブジェクトの構造からクラスを定義する方法もある。後者は、人が試行錯誤して正しい解を模索する方法に近い。

既存の言語において Harmonia はクラス，インスタンス，メソッドの動的変更に関してほぼ機能が揃っているが、いずれもクラスからインスタンスへの概念に基づくもので、他の概念を実現する機能はない。ABCL/R は ABCL/1 にリフレクション機能を付加した言語で、メタレベル操作によりオブジェクトの動的変更を実現している。ただし ABCL/R にはクラスのような複数のオブジェクトをまとめて扱う概念が十分でなく、多様なオブジェクトの概念の実現は困難である。CLOS ではメタレベル操作が開放されており、これを用いて多くの概念を実現できる。

以上述べたように、知能処理においてはオブジェクトに関して様々な概念があり、言語システムは標準的なオブジェクトシステムの他に、必要に応じて他の概念を実現する機能も持つことが重要である。

## 4.並列性と動的変更の干渉

オブジェクトの並列機能と動的変更機能を同時に使用すると、クラスのような共有データへの変更を伴うアクセスで、次の２つの場合に問題が生じる。

(1) 例えば２つのオブジェクトが、同時にクラスの変更を行ったとき。

(2) 例えばあるオブジェクトがクラスへアクセス中に、他のオブジェクトがクラスの変更を行ったとき。

(1)の場合はクラスの変更中は他の変更を許さなければよいが、(2)の場合はそれだけでは不足である。図2を例にとって詳しく見る。

図2においてクラス1のスーパークラスはクラス2、オブジェクト1と2はクラス1のインスタンスとする。

①オブジェクト1が method-1 を受け取る。method-1 はクラス1で定義され、非同期に method-2 を呼ぶ。

②次にオブジェクト2がクラス1のスーパークラスを3に変えるメソッド change-super を受け取る。

このとき次の2つの問題が生じることが考えられる。

(2-1) システムは動作不能になる。

(2-2) システムは動作するが、結果が不定になる。

## 5.干渉解決のための考察

(2-1)の原因は計算システムの実行ステップの大きさにある。メソッド検索が複数のステップに分かれ、間にクラスの変更が行われると動作不能になる可能性が生じる。

これを防ぐ1つの方法として、クラスなど共有データへのアクセスを1ステップで行うことが考えられる。クラスもオブジェクトの場合には、アクセスメソッドをひとまとめのものにする方法が考えられる。

しかしそれだけでは(2-2)は解決しない。上の例で method-1 の検索に排他制御を入れても method-1 から method-2 が呼ばれる間にクラス階層の変更があると、別のクラスの method-2 が呼ばれる可能性がある。これに関しては次のような方法が考えられる。

(a) method-2 が実行される時点での定義を用いる。

(b) 履歴を保存し、システムの環境を method-1 が実行された時点での定義に戻す。

(c) method-2 の実行を待ちスーパークラスを変更する。

(a) は一番簡単だが、全てをプログラマの責任にすることになり疑問が残る。(b),(c) は実行結果は分かりやすいが、(b) ではどこまで環境を戻すか、(c) ではどこまで待つか、いずれも“一区切りの仕事”を特定するのが困難である。

我々は(2-2)の問題に関してどの様にするのが一番自然で分かりやすいかとその実現方法を考察中である。

## 6.おわりに

オブジェクト指向言語に対して、知能処理に必要な機能として、並列性とオブジェクトの動的変更を取り上げ、どの様な機能が必要か、既存の言語では何が実現されているかを述べた。また2つの機能を同時に実現した場合の問題点を分析し、その解決法を考察した。

＜謝辞＞

本研究を行うにあたって、日頃からご指導いただき様々な助言を下さった日比野主幹研究員に感謝いたします。

参考文献

[1]尾内，鶴岡：動的オブジェクト指向プログラミングの提案と評価，電子情報通信学会論文紙D Vol.J71-D No.12 pp.2543-2554

[2]A.Yonezawa, E.Shibayama, T.Takada and Y.Honda: Modeling and Programming in ABCL/1, in "Object-Oriented Concurrent Programming" ed A.Yonezawa and M.Tokoro, MIT Press, pp.55-89(1987)

[3]T.Watanabe and A.Yonezawa: Reflection in an Object-Oriented Concurrent Language, OOPSLA'88 Proceedings pp.306-315

[4]D.Bobrow, K.Kahn, etc.: Common Lisp Object System Specification, SIG PLAN NOTICES 23-9 1988

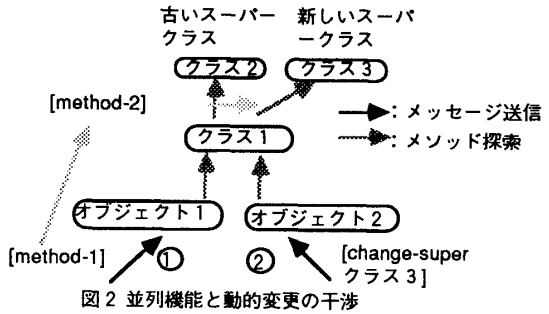


図2 並列機能と動的変更の干渉

| | Harmonia | ABCL/1(ABCL/R) | CLOS |
|---|---|---|---|
| 研究者 | 尾内,鶴岡(NTT) | 米沢,柴山(渡部,米沢)(東工大) | ——— |
| 並列機能 | あり | あり | なし |
| オブジェクト間の通信機能 | メイルボックスによる基本通信機能のみを提供 | 現在型(非同期送信)/過去型(同期送信)/未来型(非同期交信)によるメッセージ通信 | ——— |
| オブジェクトの動的変更機能 | あり | あり | あり |
| 様々なオブジェクトの概念の実現 | システムに最初に組み込まれた概念だけ | リフレクション機能により他の概念を実現可能<br>ただし、複数のオブジェクトをまとめて扱う機能が不十分 | メタオブジェクトプログラミングにより、他の概念を実現可能 |

図1 既存言語の機能