3G－7

コンカレントプログラミングのための オブジェクト機能モデル*

菅谷徹　宮寺庸造　近谷英昭

東京電機大学†

1　はじめに

コンカレントプログラミングにおいて，並列動作可能なオブジェクトを使用することはプログラムの記述能力，理解性を高め，ひいては生産性を高めるものとなる．しかし，現状では並列動作するオブジェクトを提供しているモデル，言語は少ない．

本研究は，ABCM/ABCL（及びその後継モデル，言語）[1, 2, 3] を拡張したより自然なプログラム表記のためのオブジェクト機能モデルと，汎用性のある記述言語の設計を目標とする．本稿ではオブジェクト機能モデルについて述べる．

2　オブジェクト

多くの出回っているオブジェクト指向言語では，オブジェクトの動作が並列に行われず，単に各オブジェクトのメソッドを直列に動作させるだけのものが多い．

これに対し，我々の提案するオブジェクトモデルは各オブジェクトにオブジェクトマネージャ(Manager) と呼ばれる特殊なオブジェクトを取り付けたモデルである．オブジェクトマネージャは種々のメッセージの処理や，状態情報の即答等を行い，オブジェクトと同じデータを参照でき，特定のメソッドを持つ特殊なオブジェクトである．マネージャに対して，オブジェクト本体をボディー (Body)と呼ぶ．マネージャもオブジェクトであるので，ボディーに対して独立して動作する．これにより，作業の並列度をより高めることが可能となる．本モデルのオブジェクトの構成要素を以下に示す（図1）．

・ボディー（データ，メソッド，計算パワー (CPU)）
・メッセージキュー
・オブジェクトマネージャ
　（データ，メソッド，計算パワー）
・マネージャメッセージキュー

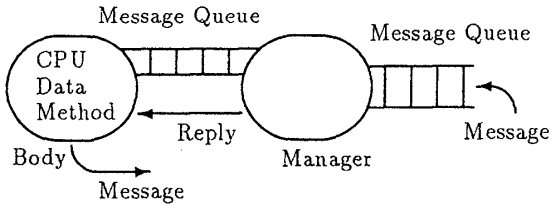


図1．オブジェクトモデル

また，我々のオブジェクトはその生成から消滅までの間に，以下の図のような4つの状態のいずれかをとる．

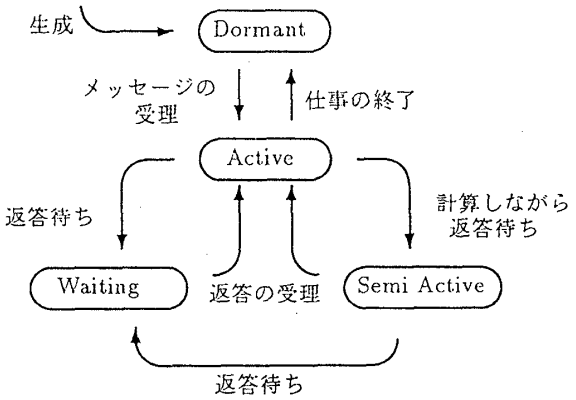


図2．オブジェクトの状態遷移図

オブジェクトは生成された時と何も作業をしていない時は休眠状態である．ここで何かメッセージを受け取ると活動状態に入り，指示された作業を行う．作業が終了すれば，休眠状態に戻り次のメッセージの到着を待つ．もし，作業中に他のオブジェクトに対して，作業を依頼すると待ち状態か準活動状態のいずれかになる．待ち状態となるのは返答をすぐに必要とする場合である．この場合，返答が到着するまで，オブジェクトの動作は停止する．返答をすぐに必要としない場合は準活動状態となる．これは，メッセージを発した後，作業を続け必要になったところで，返答の到着を確認する．到着していればそれを用いて作業を続け，到着していなければ待ち状態となる．

3　メッセージ

3.1　メッセージ形態

メッセージとその送受信は非常に重要な機構である．

メッセージには〝どのように〟を指示する必要がなく〝何をするのか〟を記述するだけですむ．よって，オブジェクトのモジュラリティの向上等機能的に非常に優れている点がある．しかし，ここで重要なのはメッセージによる送信が我々のコミュニケーション形態に非常に近いという点である．オブジェクトへのアクセスは，このメッセージ送受信を用いてのみ行われる．

本モデルのメッセージとは以下のプロトコルに沿って記述される，一連のスクリプトである．

*An Object Model for Concurrent Programming.
Tetsu Sugaya, Youzou Miyadera, Hideaki Kon'ya
†Tokyo Denki University

・Receive Object ：送信先オブジェクト
・Message Name ：メッセージ名
・Message Owner ：メッセージの発信元
・Message Sender ：メッセージの発信オブジェクト
・Arguments ：参照引数

本稿で述べるメッセージ送信は，関数呼出的ではない．よって，返答もまた明示的にメッセージ送信として記述する．

3.2 メッセージ送受信

ここでは，オブジェクトへのアクセス，すなわちオブジェクト間のメッセージの送受信について述べる．

メッセージは以下のような記述で各オブジェクトに送られる．

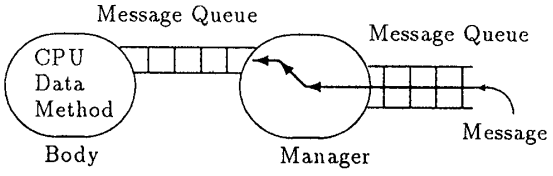
Objects ← Message: Arguments.

これにより，Object に対して Message というメッセージが送られる．引き数 (Arguments) はあってもなくても良い．以下，この形式をメッセージ式と呼ぶ．

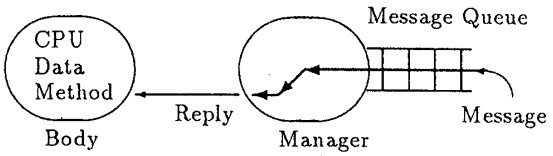
送信の際，メッセージに Message Owner と Message Sender が付随され，メッセージプロトコルが完成する．Message Owner は大元のメッセージの発信オブジェクトであり，Message Sender はメッセージを直接送信したオブジェクトである．

例えば，Object A はある作業を依頼するために Object B にメッセージを送り，Object B はその作業中にさらに Object C にメッセージを送るいう流れを仮定する．この時，Object B は Object C の返答を必要とせず，かつその返答がそのまま Object A への返答になる場合は，Object C から Object A に返答を直接送った方が効率が良い．しかし，Object A, Object C はお互いを知らないので，通信ができない．そこで，常にメッセージ内に作業の発注主を登録する．これが，Message Owner である．返答は，常にメッセージ内の Message Owner に対して行う．これにより，効率の良い返答が可能となる．

受理したメッセージはまずマネージャのメッセージキューに入り，マネージャによって処理される．マネージャはオブジェクトの状態によってメッセージを仕分けする (図 3)．オブジェクトが活動状態，休眠状態の時はメッセージはボディーのメッセージキューに送られる．待ち状態，準活動状態の時はボディーのメッセージキューには送られず，そのままボディーに送られる．



a) 通常のメッセージの受理（活動状態，休眠状態時）



b) 返答の受理（待ち状態，準活動状態時）

図 3. オブジェクト内のメッセージの流れ

本モデルにおけるメッセージは 2 種類存在する．

ひとつはメッセージを一方的に発した後，作業を続けるもので通常型 (Normal Type) と呼ぶ．これは，返答を全く必要としない場合や返答結果を即座に必要としない場合に用いられる．

もうひとつはメッセージを発した後，作業を止めて返答を待つもので，待機型 (Wait Type) と呼ぶ．これは，依頼した返答結果を受けて，作業を続ける場合に用いられる．

実際には，メッセージは通常型のみで待機型は返答を待つという，自分自身への指示により実現される．返答が，作業の終了を表さないことに注意されたい．これにより，返答を行った後で処理の継続ができるため高い並列性が期待できる．

4 並列動作

前述の通り，各オブジェクトは計算パワーを持っているため，独立して動作でき，また同じ構造を持つマネージャがメッセージの処理を行うので，主な仕事に専念できる．

また，複数のオブジェクトに対して，同時に同じメッセージを送ることや，複数のオブジェクトから同時に返答を受けることも可能である．

(r1, r2, r3) = (O1, O2, O3) ← Calc.

これは，3 つのオブジェクト O1，O2，O3 にメッセージ Calc を送り，その結果をそれぞれ r1，r2，r3 で受理することを表している．これは内容的には以下の記述に等しい．

r1 = O1 ← Calc.
r2 = O2 ← Calc.
r3 = O3 ← Calc.

異なるオブジェクトに，異なるメッセージを同時に送る記述はできないが，通常型メッセージの場合は，メッセージを送った後に，次の送信を行うので大きな違いはない．

5 まとめ

本稿では，コンカレントプログラミングを目指したオブジェクト機能モデルの提案した．

このモデルはまだ開発段階にあるモデルであり，今後多くの問題に適用しながら，改良していく予定である．

また，現在，本稿で提案したモデルの記述言語も設計している．

参考文献

[1] 柴山悦哉，米澤明憲，モデルと表現，岩波講座ソフトウェア科学 17，岩波，1992.

[2] T.A.Budd, Object-Oriented Programming, Addison Wesley, 1991.

[3] 米澤明憲，オブジェクト指向プログラミングについて，コンピュータソフトウェア，Vol.1, No.1, pp29-41, 1984.

[4] 米田信夫，プログラム言語，岩波講座情報科学 9，岩波，1983.