北海道大学工学部研究報告
第151号（平成2年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No. 151(1990)

メタレベル計算の機構を導入した並列オブジェクト指向計算モデル

繁田　良則　　渡辺　慎哉　　宮本　衛市

（平成2年4月6日受理）

An Object-Based Concurrent Model with Meta Level Computation

Yoshinori SHIGETA, Shin-ya WATANABE and Eiichi MIYAMOTO

(Received April 6, 1990)

Abstract

We propose an object-based concurrent computation model with meta level computation based on the 'Kamui88' model. This model has two kinds of meta level computation called Local and Global. Local meta level computation is where a single object is operated on, and Global meta level computation is where the relationships between objects are operated on.

We also present some applications of our model.

1．はじめに

近年のハードウェア技術の進歩にともない，並列処理や分散処理あるいはこれらを統合した並列分散処理への関心が高まってきている。こうした中で多数の並列分散処理計算モデルが提案されているが，最近，並列オブジェクト指向計算モデルが注目を浴びている。この計算モデルはオブジェクト指向計算モデル（Object oriented computational model）に基礎を置いたモデルである。このモデルでは計算の単位がオブジェクトであり，オブジェクト間の相互作用によって計算が進行する。ここでオブジェクトの実体は，内部状態（データ）とこれを操作する手続きをまとめたものである。オブジェクトの内部状態を外部からは直接操作できない。これによってオブジェクトの独立性が非常に高くなることがこのモデルの利点である。

大規模なソフトウェアの開発において，問題を複数の副問題（サブゴール）に分割し，その副問題をプログラムモジュールとして実現するという手法の有効性は広く一般に認められている。また，このモジュールの独立性が高ければ高いほどソフトウェアの信頼性および保守性が高まることも知られている。このモジュールの考えを進めたものがオブジェクトであり，その高い独立性がソフトウェア全体の信頼性に貢献する。さらに，オブジェクト間の相互作用に基づいた処理形態は，実世界とのアナロジーとして人間の思考形態とも適合している。

オブジェクトの独立性，人間の思考形態に近いモデルというオブジェクト指向計算モデルの利点に着目して，オブジェクトを並列実行の単位として捉え直すことが並列オブジェクト指向計算

情報工学科　言語情報工学講座

モデルの基本的考え方である。これまでの並列オブジェクト指向計算モデルでは，オブジェクト間の相互作用の形態（通信形態）に議論が集中しがちであった。このためオブジェクト自体に関する問題が余り議論されてこなかったが，最近ではオブジェクト自体の記述力の向上を意識した提案がなされるようになってきた。[1)2)3)]

本研究もこの一環であり，並列オブジェクト指向計算モデルに meta level computation（以下メタレベル計算あるいはメタレベル操作と呼ぶ）の機構を導入し，オブジェクトの記述力向上を目標としている。また，本研究は並列計算モデル Kamui88[4)]（以下 Kamui モデルと呼ぶ）に基礎を置き，このモデルを拡張してメタレベル計算の機構を導入し，新しい並列オブジェクト指向計算モデルを提案するものである。

2．並列オブジェクト指向計算モデルとメタレベル計算

この章では，並列オブジェクト指向モデルおよびメタレベル計算について説明を行った後，並列オブジェクト指向モデルへのメタレベル計算の導入について述べる。

2.1 オブジェクト指向計算モデル

オブジェクト指向計算モデルの最大の特徴は，問題解決の世界がすべてオブジェクトから構成されるとする考え方である。ここでいうオブジェクトとは問題領域の一部を取り出し，それを抽象化したものである。人間が問題を解決する場合，問題を整理していくつかの小問題に細分化することが通常行われる。これと同様にオブジェクト指向モデルでは問題領域を抽象化された実体であるオブジェクトで分割し，問題を表現する。これによって問題を自然な形（人間の思考に近い形）で表現しようとするのである。

実際のオブジェクトは，

・内部状態（データ）

・内部状態を操作する手続き群

から構成され，内部状態の操作は手続き群を介してのみ行うことができる。また，実際の計算はオブジェクト間のメッセージのやり取り（メッセージパッシング）によって行われる。

2.2 並列オブジェクト指向計算モデル

並列処理のプログラミングを考える上で最も重要なことは，いかに自然に並列な動作を記述できるかということである。人間の並列思考能力は意外に小さいものである。したがって，あるまとまった処理は従来どおり逐次的に記述し，これらのかたまりが並列に動作しているという形態が人間にとって最も自然であろう。

このかたまりのひとつをオブジェクトとして捉えることが並列オブジェクト指向計算モデルの基本的な考え方である。また，オブジェクトの独立性とメッセージパッシングという相互作用の形態が，実世界のアナロジーとして人間の思考に沿った問題の自然な記述へとつながる。一方，並列オブジェクト指向計算モデルでは，オブジェクト間の相互作用がメッセージパッシングという疎な結合であるため，多数のコンピュータにまたがった分散処理環境への拡張も比較的容易である。これが並列分散環境での計算モデルとして，並列オブジェクト指向計算モデルに期待が寄せられている理由である。

2.3 メタレベル計算とは

まず，計算システムについて考えてみる。計算システムとは，対象とする問題領域の現象や動作をモデル化し，プログラムとデータで表現したものである。対象となる問題は物理系のように具体的なものであったり，もっと抽象的なものであったりするが，特に対象が別の計算システムである計算システムをメタ計算システムと呼ぶ(図2.1)。そして，メタ計算システムにおける計算をメタレベル計算といい，そのときのプログラミングをメタレベルプログラミングと呼ぶ。メタレベル計算の基本的考え方は，"従来混然一体となって表現されていた問題を対象レベルとメタレベルの観点から分離し，対象レベルの計算に関するメタな情報をメタレベルで明示的に記述する"[5]というものである。これによって計算システムやプログラミング言語に柔軟な計算メカニズムの枠組みが提供されることになる。

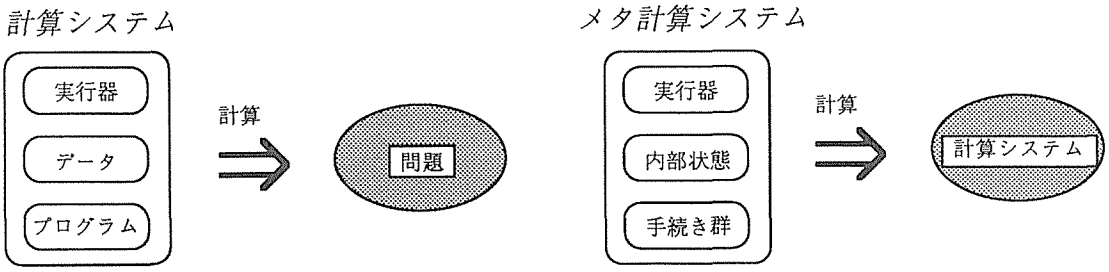


図2．1 計算システムとメタ計算システム

2.4 並列オブジェクト指向計算モデルにおけるメタレベル計算

並列オブジェクト指向計算モデルでは，その独立性を理由にオブジェクトをメタレベル計算の単位とすることが一般的である。[2)3)] これは，オブジェクトが比較的他とは独立した存在であるので，オブジェクトに対するメタレベル計算もオブジェクトを単位として行うものとする考え方である。具体的には，図2.2 a)のようにメタオブジェクトとオブジェクトが対をなしていて，このメタオブジェクトによってオブジェクトの状態がコントロールされるのである。しかしこの方法では，メタオブジェクトは管理しているオブジェクトの局所的な情報を扱うことはできるが，オブジェクト間にまたがる大域的な情報を扱うことはできない。小規模な問題であれば，この方法でも有効であるが，多数のオブジェクトが存在し，相互に作用を及ぼし合っているような場合には，オブジェクト間の大域的な情報を扱える方が，問題に対する記述力は向上する。

次にオブジェクト間の大域情報を扱うために図2.2 b)のように，メタオブジェクトを共有する形態について考えてみる。この方法を用いれば，確かにメタオブジェクトを介して大域的情報を扱うことができるが，オブジェクトの数が多くなるとメタオブジェクトに全ての情報が集中し，分散処理という並列オブジェクト指向モデルの利点が失われてしまう。

そこで本論文では，図2.2 c)に示すメタレベルの操作をオブジェクトごとの局所的情報の操作とオブジェクト間の大域的情報の操作の2つに分離したモデルを提案する。このモデルでは，

・オブジェクト間の大域的情報はメタ系が扱う

・オブジェクトごとの局所的情報はメタオブジェクトが扱う

ものとする。このように役割を分担することによって，共有メタオブジェクト型で問題であったメタオブジェクトへの情報の集中も緩和され，かつオブジェクト間の大域的情報も扱えるようになる。本研究では，Kamuiモデルの"場"を拡張してメタ系を導入する。

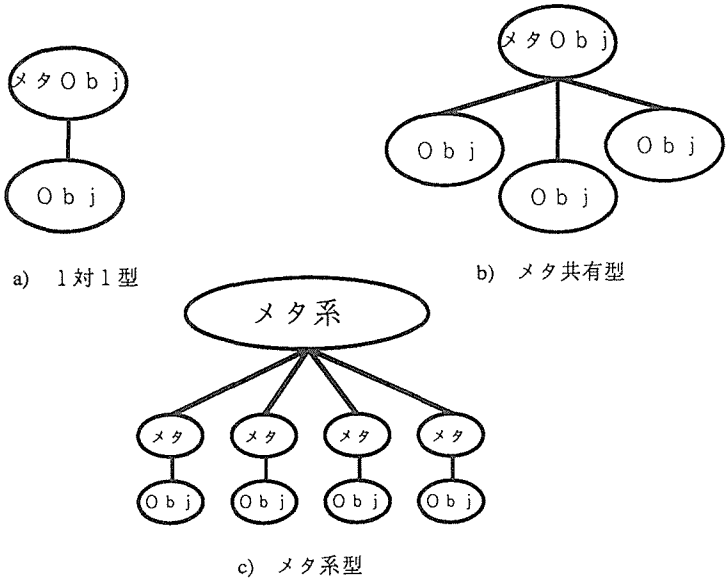


図2．2 メタ計算システムの形態

3．メタレベル計算機構の導入

この章では，Kamuiモデルを見直しその問題点を挙げた後，メタレベル計算機構を導入した新しい計算モデルMeta level Kamuiモデル（以下Mt. Kamuiモデルと呼ぶ）について説明する。

3.1 Kamuiモデル

ここでは，Kamuiモデルの構成要素である"場"と"Kamuiオブジェクト"について説明する。

(1) 場

場はKamuiモデルの最大の特徴であり，オブジェクト群をグループ化するものである。場の基本的な動作は，場に対して送られたイベント（メッセージ）をその場に属するすべてのオブジェクトにブロードキャスト（放送）することである。この機構によって，Kamuiモデルにおけるオブジェクトは相手を明示的に指定しないで通信を行うことが可能となり，より高い独立性が得られるのである。また，各オブジェクトは複数の場に属することができ，場への出入りは動的に行うことができる。さらにオブジェクトは場を生成することができる。

(2) Kamuiオブジェクト

Kamuiオブジェクトは，

・内部状態（データ）
・行動記述部
・イベントキュー

から構成されている。オブジェクトの内部状態は局所的なものであり，行動記述部でのみアクセスが許される。行動記述部はイベントに対応したオブジェクトの行動を記述する部分で，アトミックであることがモデルによって保証されている。また，オブジェクトに送られてきたイベントは，到着順にイベントキューに入れられる。

オブジェクトの基本動作はイベントキューの先頭からイベントを1つ取り出し，イベントに対応する行動記述部を実行することである。イベントに対する行動記述部が存在しない場合には，そのイベントは単に捨てられる。

3.2 Kamuiモデルの見直し

ここでは，オブジェクトのグループ化と実行環境の観点からKamuiモデルを見直し，メタレベル計算機構導入の必要性およびメタレベル計算の大域的，局所的分割の有効性について述べる。

(1) オブジェクトのグループ化

Kamuiモデルにおけるオブジェクトのグループ化は弱いグループ化であると言える。なぜならKamuiモデルではイベントの配送範囲限定の手段としてのみオブジェクト群のグループ化を捉えているからである。このことは場の動作を考えれば分かる。場は送られてきたイベントを，その場に属しているすべてのオブジェクトに再配送するだけである。場はその場に属しているオブジェクトのリストを持っているだけで，その他の情報は一切持っていない。このような弱いグループ化はある程度有効であるが，オブジェクトのグループ化の利点を生かしきっているとは言えない。また，逆に強過ぎるグループ化はオブジェクト間の相互作用を制限することにつながり，オブジェクト指向モデルの利点であった個々のオブジェクトの独立性と自由度を奪う危険性がある。

そこでMt. Kamuiモデルでは，Kamuiモデルよりも強いが，オブジェクトの独立性と自由度を奪わない程度の強さのグループ化を考える。具体的には，場がその場に属しているオブジェクト群に関する情報とその情報を操作する手続き群を持つことができるようにするのである。この拡張した場を，Mt. Kamuiモデルではフィールド(Field)と呼ぶことにする。(以下"場"はKamuiモデルの場を，"フィールド (Field)" はMt. Kamuiモデルの場を指すことにする) このとき，フィールドにおける計算は，そのフィールドに属しているオブジェクト群を対象にしていることより，オブジェクト群に対するメタレベル計算であるといえる。

また，フィールドが持つべき情報としては，

- フィールドに属しているオブジェクトの数
- オブジェクト群が受理するイベントの種類
- オブジェクと群に関する統計情報

などが考えられる。

(2) オブジェクトの実行環境

Kamuiモデルのオブジェクトの動作は，イベントキューからイベントを一つ取り出し，そのイベントに対応する行動記述部を実行するというものである。このような動作形態を取ることによって，プログラムはイベントと行動を列挙するだけで良くなり，その可読性が向上する。しかし，イベントの選択的な受理を行いたい場合，すなわちイベントの受理順序が計算結果に影響を与える場合には，イベントと行動を列挙するプログラミングスタイルでは行動の記述が煩雑になりがちである。そこで従来のオブジェクトをイベントの選択的受理を行うオブジェクト(Obj-A)とオブジェクト本来の動作を行うオブジェクト(Obj-B)とに分離することを考える。こうすることによってオブジェクトの枠組が明確になり，プログラムの可読性が向上すると共に，問題に対する自然な記述が可能となる。ここで，Obj-Aにおける計算はObj-Bを対象としていることになる。この意味においてObj-Aにおける計算はObj-Bに対するメタレベル計算であると考えるこ

とができる。

一般に，オブジェクトを計算対象とするオブジェクトはメタオブジェクトと呼ばれ，対象となるオブジェクトは単にオブジェクトと呼ばれるが，Mt. Kamui モデルではオブジェクトの性質をはっきりさせるために，

・メタオブジェクトをマネージャオブジェクト（Manager object）

・通常のオブジェクトをワーカオブジェクト（Worker object）

と呼ぶことにする。また，マネージャオブジェクトはワーカオブジェクトのイベントキューだけでなく，内部状態など実行環境のすべてを計算対象としている。

3.3 Mt. Kamui モデル

ここでは，Mt. Kamui モデルの構成要素についての説明を行う。

(1) 構成要素

Mt. Kamui モデルの構成要素は以下の3種類のオブジェクトである。

・フィールドオブジェクト（以下 Field とする）

・マネージャオブジェクト（以下 Manager とする）

・ワーカオブジェクト（以下 Worker とする）

図 4.2 c）において，メタ系が Field に，メタオブジェクトが Manager に，通常のオブジェクトが Worker に対応している。また，Field, Manager, Worker はオブジェクトであるので，内部状態とそれを操作する手続き群を持つことになる。

(2) Field

Field は次の3つの手続きを必ず持っている必要がある。

・enter-field Field にオブジェクトが入る時に呼び出される

・exit-field Field からオブジェクトが出る時に呼び出される

・send Field にイベントが到着した時に呼び出される

これらの手続き以外にも，オブジェクト群を操作するための手続きを持つことができる。

(3) Manager

Manager は次の手続きを必ず持っている必要がある。

・send Manager が管理している Worker にイベントが到着した時に呼び出される

この手続き以外にも，Worker を操作するための手続きを持つことができる。

(4) Worker

Worker は Field や Manager のような特定の手続きを持つ必要はない。また，Worker の内部状態や手続き群あるいはイベントキューは，Manager の内部状態として実現されることになる。

3.4 例題

(1) イベントの選択的受理 1

いま，ある条件が成立した場合のみイベントを受理するオブジェクトを考える。Kamui モデルでは，これを条件が不成立の間はイベントを自分のイベントキューの最後に戻すことによって実現し，受理の先送りを行う。PCL（Portable Common Loops）の表記法を用いると，次のようになる。

```
(method 1 ( )
    (cond ((条件が不成立) event-send self 'method1))
          ((条件が成立) (行動 1))))
(method 2 ( )
    (cond ((条件が不成立) (event-send self 'method 2))
          ((条件が成立) (行動 2))))
```

Kamui モデルでは，それぞれの method 内で条件が不成立の場合の処理を行わなければならない。そのため処理の表現が煩雑となり，可読性も低下している。これに対して Mt. Kamui モデルではイベントの選択受理は Manager が一括して行うため，Worker は行うべき本来の処理を以下のように自然に表現することができる。

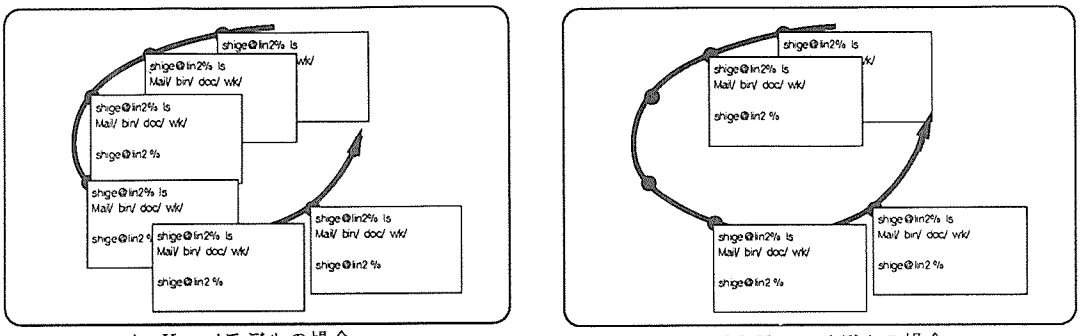
```
;;; Manager
(cond ((条件が不成立) (何もしない))
      (条件が成立) (Worker にイベントを送る))

;;; Worker
(method 1 ( ) (行動 1))
(method 2 ( ) (行動 2))
```

(2) イベントの選択的受理 2

ウインドウシステム上のウインドウをドラッグしている場合を考える。ウインドウシステムのサーバからは，マウスの位置の変化がイベントとして逐次送られてくるものとする。ウインドウの再描画が素早く行える場合には，マウスの位置とウインドウの位置がずれることはないが，マウスの移動速度に比べて再描画に時間がかかる場合には，マウスの位置とウインドウの位置はずれことになり，イベントキューには再描画のイベントが大量に貯まることになる。しかし，この場合意味のあるイベントは最新のマウスの位置だけであり，それ以外のイベントはなんら意味をもたない。

Kamui モデルではオブジェクトは，イベントキューの先頭からイベントを取り出す以外のことはできず，サーバから送られてきた再描画のイベントをすべて処理しなければならない。したがって，再描画に時間がかかる場合には，マウスの位置とウインドウの位置がずれることになる。これに対して，Mt. Kamui モデルではイベントキューがマネージャオブジェクトの内部状態として実現されるため，マネージャオブジェクト内で最新の再描画イベントだけを取り出し，古いイベントを捨てることができるので，マウスの位置とウインドウの位置がずれることはない。



a) Kamuiモデルの場合　　b) Mt.Kamuiモデルの場合

図3．1　ウインドウのドラッグ

4．Mt. Kamui モデルに基づいた言語処理系の実現

Mt. Kamui モデルに基づいた言語処理系を PCL 上で実現した。Mt. Kamui モデルは並列オブジェクト指向計算モデルであるので，本来ならば各オブジェクトは並列に動作すべきであるのだが，プロトタイプとして擬似並列に動作する処理系を作成した。

本言語処理系には，問題を記述し実際に動作させるために，

- defField　　Field を記述する
- defManager　　Manager を記述する
- defWorker　　Worker を記述する
- mtk-send　　イベントを送る
- mtk-send-meta　メタレベル操作としてイベントを送る

が用意されている。例として，Kamui モデルの "場" の動作をシミュレートする Field の記述を示す。

```
(defField Standard-Field
  (Inherit ())
  (Slots ((queue nil) (self nil) (entry nil)))
  (Script (start-up (me)
            (setq queue (make-instance 'Queue))
            (setq self me))
          (enter-field (obj)
            (setq entry (adjoin obj entry)))
          (exit-field (obj)
            (setq entry (set-difference (list obj) entry)))
          (send (event)
            (en-que que event))
          (perform ()
            (unless (empty-p queue)
```

```
        (let (event (de-que queue)))
            (mapcar #' (lambda (x) (mtk-send x event)) entry))))
))
```

Inherit は他の Field の記述を継承するためのものであり，Slots は内部状態を，Script は内部状態を操作する手続き群を定義するものである。また，Slots の queue, self entry はそれぞれイベントキュー，自分自身の ID，場に属しているオブジェクトのリストである。

5．おわりに

本研究の主題は，並列オブジェクト指向計算モデルへのメタレベル計算の導入と，メタレベル計算をオブジェクトごとの局所的なものとオブジェクト間にまたがる大域的なものに分割することであった。大域的メタレベル計算を Kamui モデルにおける"場"の拡張として捉えている点が本研究の特徴である。今後，メタレベル計算を一歩進めた自己反映計算[1)3)5)]についての考察を行っていく必要があると思われる。

本研究を行うにあたり，御指導，御助言を頂きました北海道大学工学部　情報工学科　赤間清助教授，三谷和史助手に深く感謝致します。

文　献

1）Smith, B.C. : Proc. 11th ACM Symposium on Principles of Programming Language (1989), pp. 22－35.
2）T. Watanabe, A. Yonezawa : Proc. of OOPSLA, ACM September 1988, pp. 306－315.
3）渡部，米澤：情報処理学会研究報告 89-SF-33，89-PL-23 (1989)，pp. 77－85.
4）渡辺　他：コンピュータソフトウェア，Vol. 6，NO. 1 (1989)，pp. 41－55.
5）管野，田中：情報処理 Vol. 30，No. 6 (Jun. 1989)，pp. 694－705.