オブジェクトと場に基づいた協調的プログラム言語

西尾郁彦† 渡辺豊英† 杉江昇†

複数の実体が特定の目的や関係にしたがって集団を形成し，互いに協調して仕事を行うという状況は非常に一般的な実体の活動形態である．しかし，このような実体の集団活動をオブジェクト指向モデルによって表現する場合，いくつかの問題が生じる．オブジェクト指向モデルでは，オブジェクトが集団を形成するための機構にモデル化の視点がおかれていないからである．本論文では，集団を形成したオブジェクト間の協調動作を表現するための枠組みをオブジェクト指向パラダイムに基づいて構成する．集団活動の表現には次の事柄が重要である．(i)集団内では不特定多数のオブジェクト間での間接的な通信が必要となる．(ii)集団もまた一つのオブジェクトとしてとらえなければならない．これらの要求に対して，オブジェクトの活動環境，通信媒体としての「場」を導入し，さらに場によって定められるオブジェクトの集団を一つのオブジェクトとして抽象化して扱うモデル化の手法を提示する．本モデル化の特徴は，集団内のオブジェクトの相互作用，および集団と他のオブジェクトの相互作用という集団が持つ二つの異なる視点を，場の概念，集団オブジェクトの概念によってそれぞれのレベルからモデル化することができる点にある．また，本モデル化の枠組みに基づいた実験的なプログラム言語についても言及する．

A Cooperative Programming Language Based on Objects and Fields

FUMIHIKO NISHIO,† TOYOHIDE WATANABE† and NOBORU SUGIE†

The object-oriented model is very successful to represent various phenomena as entities in the real world. However, in case of modeling the group activity of these entities in the object-oriented model, some restriction arises. The reason is that the object-oriented model focuses on individual objects, but not on the group of objects. In this paper, we show a framework to model the cooperative activities of objects based on the object-oriented paradigm. The following subjects are important to represent the group activities systematically: (i) The indirect communication method is needed for many unspecified objects in a group; (ii) A group is looked upon as an abstract object constructed by several related objects. Therefore, we introduce the concept of field, which is a communication environment for objects, and the group of objects is modeled not only as a field but also as an object in accordance with our different modeling views. We also show our experimental programming language based on this modeling method.

1. はじめに

オブジェクト指向パラダイムは実世界の実体を自然な形でモデル化することができ，情報システムの構築に大きな影響を与えている[1]．また，このパラダイムに基づいて Smalltalk-80[2]，C++[3]などさまざまなオブジェクト指向言語が開発，利用されている．オブジェクト指向パラダイムでは，実体をオブジェクトとしてモデル化し，オブジェクト間の相互作用を通して問題を表現する．また，実世界では実体は並行に存在し，活動することにより，それぞれが独立に個々の処理を実行している．オブジェクトの独立性，並行性に注目し，並行プログラミングの機能をもたせた並行オブジェクト指向言語もいくつか提案されている[4),5]．ABCL/1[6]はアクタに基づいた並行計算モデルであり，並行プロセスとプロセス間の通信を，並行オブジェクトとオブジェクト間のさまざまな相互作用によって表現する言語である．これらの言語では，オブジェクトを現実世界における実体と1対1に対応させ，それを実行の単位としている．これにより，並行オブジェクト指向言語では自然に並行プロセスを記述することが可能となる．

並行プロセスを自然に記述できることから，人工知能の一分野として近年研究されている分散 AI に，これらの言語を適用する試みも報告されている．Orient 84/K[7]は知識表現，知識処理システムを記述するための並行オブジェクト指向言語であり，オブジェクトそ

† 名古屋大学工学部情報工学科
Department of Information Engineering, School of Engineering, Nagoya University

れぞれが知識をもち，並行に活動して問題を解決していく．このように，複数の並行処理実体による協調動作のモデル化，記述に並行オブジェクト指向の考え方は有用であり，さまざまな事例に適用されている[13]．

このような協調動作のモデル化にはオブジェクト間の相互作用が重要となる．オブジェクト指向パラダイムでは，各オブジェクトは独立した存在であり，これらのオブジェクト同士のメッセージ通信による情報交換によって問題を表現する．しかし，現実世界における実体の活動を考えてみると，実体は完全に独立した存在ではなく，関係を有する他の実体と集団を形成し，この集団の中で相互に作用し合って活動している．複数の実体が特定の目的，関係にしたがって集まり，それぞれの仕事を行う状況は非常に一般的な実体の活動形態である．しかし，このような実体の集団活動をオブジェクト指向モデルに基づいて表現する場合，いくつかの問題が生じる．今日報告されている多くのオブジェクト指向モデルでは，オブジェクト単体にモデル化の視点をおいており，オブジェクトの集団に関してはほとんど考慮していない．すなわちオブジェクトの集団を表す概念をもたない．そこで，オブジェクトのみに注目するのではなく，複数のオブジェクトから構成されるオブジェクトの集まりにも注目したモデル化の重要性が提案されている．従来，多くのオブジェクト指向モデルではクラス階層により，オブジェクト間に存在する汎化・特化関係を表現してきた．しかし，この関係はオブジェクト個々の性質に基づいたオブジェクトの分類であり，ある機能を果たすために複数のオブジェクトが集まって活動する場合のオブジェクトの行動に基づいたオブジェクトの集合を表現するものではない．オブジェクトの集まりがまた新しくオブジェクトを構成するという見方から，オブジェクト指向モデルをとらえた場合，汎化・特化関係とは異なった，部分・全体関係を操作する必要がある．この関係を表現するために Blake ら[8]は，クラス階層に加え，パート階層を導入した．パート階層によって，オブジェクトの集合が全体として一つの機能を果たす機構を実現した．一方，SNOOPS[9]では Manager Object という特別なオブジェクトを用意している．これによって構成要素となるオブジェクトを管理し，オブジェクトの集団としての機能を実現している．また，Helm ら[10]は，Contract によってオブジェクトの集団を明示的に定義している．これらのアプローチはいずれも，グループを管理するための要素を明示的に定義しており，管理されるグループの構成要素もあらかじめ定義されるため，グループの構成要素となるオブジェクトがグループ間を自由に移動することはない．また，部分と全体の関係の記述に注目し，全体を表すオブジェクトは構成要素となるオブジェクトを管理する管理者としての役割をもっている．そのため，パート階層によって表現できるのは階層中の親と子の間の静的な関係であり，部分となっている個々のオブジェクト間での協調動作は十分に表現できない．

このような試みに対して，Kamui-88[11]では協調動作のためにオブジェクト間の通信媒体として，場の概念を導入している．組織計算モデル[12]においても，不特定多数のオブジェクトと同時に通信するために環境という概念を定めている．環境はオブジェクトが存在する場を定め，環境を介したオブジェクト間の間接的な通信を実現する．これらは，オブジェクトとそれが活動する環境との関係を場によって達成している．さらに，グループを構成する個々のオブジェクトは場を介することにより互いに作用しあうことが可能となる．しかし，場はあくまでも通信媒体となる受動的要素であり，それを能動的な活動実体としてとらえてはいない．集団内での相互作用を場により達成しているが，場によって定められるオブジェクトの集団同士での相互作用や活動の表現は困難である．

本論文では，オブジェクト間の協調動作を表現するための枠組みをオブジェクト指向パラダイムに基づいて構成する[16]．集団を形成するオブジェクト間の相互作用のための活動環境として，場の概念を導入する．さらに，オブジェクトの集団をたんに活動環境という受動的，静的な要素としてとらえるだけでは不十分であるという考察に基づき，集団もまた一つの活動実体としてモデル化する機構を報告する．すなわち，われわれのアプローチではオブジェクトの集団を，集団を形成するオブジェクトの活動環境と，集団もまたオブジェクトとしての一つの活動実体であるという二つの側面からとらえる．また，この枠組みに基づいた実験的なプログラム言語についても言及する．

2章でモデルの概要を述べ，3章ではモデルを構成する各要素の詳細，4章では処理系の実装について報告する．5章では例題を通してその応用例を示す．また，6章では考察と今後の課題についてまとめる．

2. Object-Field モデル

オブジェクトの集団に注目したモデル化の重要性は前節で述べたとおりであるが，オブジェクトの集団活動を扱うために必要な要件として以下の事項をあげることができる．

（i） オブジェクト間の通信機能

オブジェクトは集団内の他のオブジェクトと相互に作用しあいながら，処理を進めていかなければならない．この場合，他のオブジェクトと直接通信するだけでなく，集団内の不特定多数のオブジェクトとの間接的な通信が必要となる場合がある．

（ii） オブジェクトの集団の表現

オブジェクトは集団を形成することで新たな機能単位を形成する．集団もまた他の集団と相互に作用しあって活動している．すなわち，集団を一つのオブジェクトとしてとらえることが可能である．

多くのオブジェクト指向モデルはクラスの概念をもち，これに基づいてオブジェクトの構造をメソッド，インスタンス変数という形式で記述する．このようなオブジェクト指向モデルでは個々のオブジェクトをモデル化の単位と考えており，オブジェクトの集まりやオブジェクトが存在する環境の概念がないために，オブジェクト間の直接通信が基本となっており，集団内での相互作用を表現することが困難である．そこで，オブジェクト以外の要素として新たに場の概念を導入したモデル化が試みられている．Kamui-88[11]や組織計算モデル[12]においても，場や環境という概念の導入によってオブジェクトの集団活動のモデル化を行っている．しかし，これらの方法では（ii）の要件であるオブジェクトの集団を表現することができない．

一方，（ii）の要件に対しては，Craske[9]，Helm[10]らによって部分と全体を構成するオブジェクト間の関係を表現することにより，解法が示されている．ただし，ここでは集団を表現するオブジェクトの定義中においてオブジェクトがどのような集団を形成し，どのような関係をもつかが静的に記述される．そのため，メンバが動的に変化するような集団を表現することは困難である．また，部分・全体間の相互作用に視点がおかれ，メンバ間の相互作用については表現できない．

このように（i）（ii）の要件に対しては，それぞれ別々の視点から問題が提起され，そのアプローチが報告されている．しかし，（i）（ii）の要件はそれぞれ独立に考えられるものではない．なぜならば，これらの二つの要件はモデル化する際の視点に依存している場合が多いからである．たとえば，電車と乗客の関係を考えてみる．電車に乗っている乗客の視点で考えるならば，電車は一種の場としてモデル化することができる．車内アナウンスなどは，電車という場を通して乗客に伝えられる不特定多数への間接通信である．これに対して，電車の外にいる人の視点でみた場合，車中にどのような乗客がいるかということはあまり重要ではなく，電車という一つのオブジェクトをとらえる必要がある．従来のアプローチでは（i）（ii）の要件をそれぞれ独立にとらえていたために，集団のもつ二つの側面をうまくモデル化することができないと考えられる．

われわれが提案するアプローチでは，集団のもつ二つの側面を統合してモデル化を行う．オブジェクトの集団活動に注目したモデル化のために，場の概念を導入し，これを基礎とする．さらに，集団活動を抽象化した一つの集合オブジェクトとして場をモデル化する．その特徴は次のとおりである．

（i） 集団内でのオブジェクトの相互作用に必要である間接通信を実現する．

（ii） 集団もまた一つのオブジェクトとして抽象化して扱う．これにより，集団の集団といったオブジェクトの階層的な関係を表現することができる．

場はオブジェクトが存在し，活動する環境である．ある共通の目的，関係をもったオブジェクトは同一の場において活動する．また，同じ場に属する不特定多数のオブジェクトが場を介することにより，通信し合うことができる．さらに，オブジェクトの集団は固定化されたものではなく，オブジェクトはその活動状況に応じて任意に場を移動することができる．このように，場はオブジェクトとは異なった働きをする要素であり，場がオブジェクトに対して直接メッセージを送るなどして働きかけることはない．場の概念を導入することにより，われわれのモデルは，動的で能動的なオブジェクトとしての実体と，静的で受動的な場としての実体の二つの概念に基づく，**図1**にオブジェクトと場の関係を示す．オブジェクトは生成されると必ず場に属して活動する．また，活動の状況に応じて別の場へ移動することもある．

また，場によって互いに作用し合うオブジェクトの集合を定めるだけではなく，さらにその集団をオブジェクトとして扱うことができなければならない．そ

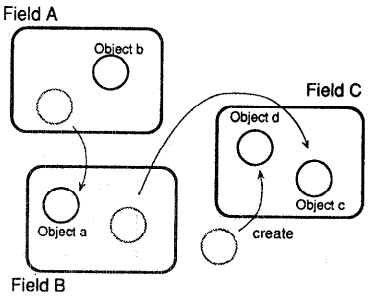


図 1 オブジェクトと場の関係
Fig. 1 Relationship between objects and fields.

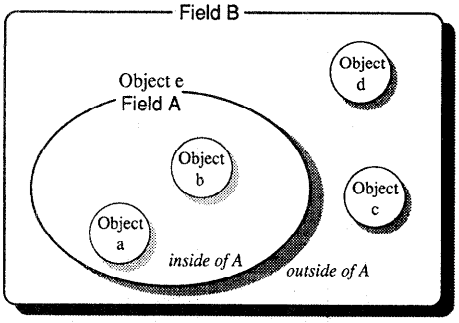


図 2 オブジェクトと場の階層構造
Fig. 2 Hierarchical structure.

こで，集団をモデル化する視点の違いに応じて場を仮想的なオブジェクトとしてとらえる．この様子を**図2**に表す．オブジェクトa，bは場Aの中で活動するオブジェクトである．さらに，a，bからなるオブジェクトの集まりはオブジェクトc，dにとってオブジェクトeとして認識され，これらは場Bの中で活動するオブジェクト群である．図2に示すように，オブジェクトの集まりを集合オブジェクトとしてとらえることにより，オブジェクトの階層的な関係を構成することができる．

3. オブジェクトと場の構造

本章では，前章で述べたモデルの構成要素について述べる．同時に，それぞれの記述形式も示す．記述言語は Smalltalk に基づいた言語である．

3.1 オブジェクト

オブジェクトの構造を**図3**に示す．オブジェクト指向モデルにおけるオブジェクトは通常，状態変数とメソッドから構成されるが，さらにこれに加え，メッセージ・バッファとメッセージ・セレクタをもつ．それぞれの役割は，次のとおりである．

- 状態変数…オブジェクトの状態を保持する変数．

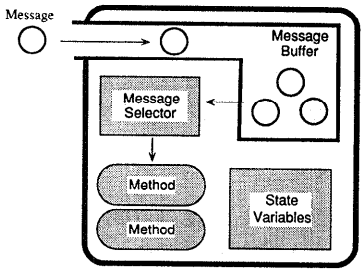


図 3 オブジェクトの構造
Fig. 3 Structure of object.

- メソッド…オブジェクトがメッセージに対してどのように振舞うかを定義した手続き．
- メッセージ・バッファ…オブジェクトに対して他のオブジェクトから送られてきたメッセージを保持するためのバッファ．オブジェクトは並行に動作しており，他のオブジェクトから非同期的に送られてくるメッセージのうち未受理のものを保持する．
- メッセージ・セレクタ…メッセージ・バッファから実行可能なメッセージを取りだし，メソッドを起動する手続き．

オブジェクトは活動実体を表現する要素であり，各オブジェクトは並行に動作している．オブジェクトが集団を形成し，その中で他のオブジェクトと協調的に動作するためには，他の多数のオブジェクトから送られてくるメッセージを処理しなければならない．このとき，メッセージは他のオブジェクトと同期的にやり取りされるだけではなく，非同期的にやり取りされなければならない．これによって各オブジェクトは独立に処理を進めることができる．オブジェクトは受けとったメッセージをすぐに受理するのではなく，いったん自身のメッセージ・バッファに格納する．メッセージ・セレクタはオブジェクトの状態等を考慮して，メッセージ・バッファに格納されたメッセージを検索し，実行可能なメッセージを取りだし，メソッドを起動する．

多くのオブジェクト指向モデルにおけるオブジェクトは，メッセージを受けとるとただちにそれを受理，実行する．つまり，オブジェクトの行動はメッセージに従属している．これに対して，われわれのモデルにおけるオブジェクトは，オブジェクト自身がメッセージのスケジューリングをメッセージ・セレクタにより自律的に行う．ここでは，オブジェクトはメッセージに対する反応体ではない．不特定多数から非同期的に

```
Entity subclass: #ObjA
   instanceVariableNames: 'i1 i2'
   classVariableNames: 'c1'
   poolDictionaries: 'p1' !

! ObjA class methods !
  < definitions of class methods > !!
! ObjA methods !
  < definitions of methods > !!
! ObjA selector: #SelectorA !
  < definition of the message selector > !!
! ObjA selector: #SelectorB !
  < definition of the message selector > !!
```

図 4 オブジェクトの記述
Fig. 4 Description of object.

メッセージが送られてくるような状況では，メッセージに対する反応の仕方はオブジェクト自身が決定しなければならない．メッセージ・セレクタの存在はこれを可能とする．

オブジェクトは状態変数，メソッド，メッセージ・セレクタの記述により定義される．記述形式の骨格を**図 4** に示す．

メッセージ・セレクタは，受けとったメッセージとオブジェクトの状態変数から適切なメソッドを実行する制御部であり，ここにはメソッド名とそれを受理するために満たすべき条件の組を記述する．さらに，メッセージ・セレクタは一つのオブジェクト・クラスに対して複数定義することもでき，オブジェクトの生成時にクラス名と同時にセレクタ名を指定することで，任意のメッセージ・セレクタをもったオブジェクトを生成できる．これによって，同じクラスから生成されたオブジェクトでも異なるメッセージ・セレクタをもつことができる．

3.2 場

場はオブジェクトの活動環境を定め，通常のオブジェクトとは異なる受動的な要素である．オブジェクトは場を介することで，同時に同一場内の不特定多数のオブジェクトと通信することができる．

場がもつ役割として，

- 場に対して送られたメッセージを，場に存在するオブジェクトに同時放送する．
- 場に存在するオブジェクトのリストを保持し，オブジェクトの場への入出に応じてリストを更新する．
- オブジェクトが活動する環境の状態を表す状態変数を保持する．この状態変数はオブジェクトから参照・変更される．

がある．これらの役割を満たすため，場は以下の要素から構成される（**図 5**）．

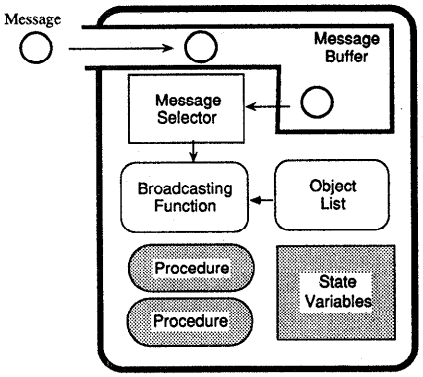


図 5 場の構造
Fig. 5 Structure of field.

- 状態変数…場の状態を表す変数．
- 手続き…場の状態変数を参照・変更するための手続き
- メッセージ・バッファ…場に対して送られたメッセージを一時的に保持する．
- ブロードキャスト機能部…場に対して送られたメッセージを，場に存在するオブジェクトにブロードキャストする．
- オブジェクト・リスト…場に存在しているオブジェクトを管理する．

オブジェクトは場にメッセージを送ることで，場に存在する他のオブジェクトと間接的に通信することができる．メッセージを受けとった場は，ブロードキャスト機能によってメッセージを同時放送する．ブロードキャスト機能は，場に存在しているオブジェクトを示すオブジェクト・リストも管理しており，これに基づいてブロードキャストを行う．また，オブジェクトの場への入出要求を受けてリストを更新する．

場は状態変数とそれを参照する手続きの記述によって定義される．その記述形式は**図 6**（a）に示すように，オブジェクトとほぼ同様の形式であり，クラス

```
(a)
Field subclass: #FieldA
   instanceVariableNames: 'i1 i2'
   classVariableNames: 'c1'!

! FieldA methods !
<definitions of methods
  to access instance Variables> !!

(b)
f := FieldA create.
f enter: anObj.
f exit: anObj.
f destruct.
```

図 6 場の記述
Fig. 6 Description of field.

Field のサブクラスとなる．ブロードキャスト機能はすべての場に共通する場の性質であるから，これを記述する必要はない．

オブジェクトは create, destruct という操作により随時場を生成，削除することができる．また，オブジェクトは生成された場に対して enter という操作で参加し，exit によって場より退出する．enter, exit の引数には自分自身を表す変数 self の他に，任意のオブジェクトを指定することができ，これによって他のオブジェクトを場に参加させたり，退出させたりする（図6(b)）．

場の概念を導入することの利点として，オブジェクトのグループを場によって表現することができ，グループ内で不特定多数の相手と通信するような状況の表現が容易になる．また，オブジェクトの場への入出によって，動的に変化するオブジェクト間の関係を扱うことも可能である．

3.3 集団としての場

3.2 節で述べたように，場は複数のオブジェクトに共通な動作環境を定め，オブジェクトのグループを表現する受動的要素である．しかし，グループをモデル化する視点によっては，それをオブジェクトと同様な能動的実体としてとらえる必要がある．このようなグループの能動的な側面を定義するために，すでに定義された場，つまりグループの受動的側面に対して新しくオブジェクトの性質を重ね合わせる．これを**図7**に概念的に示す．オブジェクト c は場 A に存在している．一方，オブジェクト b にとっては，受動的な要素としての場 A ではなく，むしろこれは b と対等なオブジェクト a であり，この二つの視点の違いを場 A とオブジェクト a の重ね合わせによって扱う．オブジェクトのグループ化に関する提案はいくつかなされてきたが，そこではグループ化機能のみに着目し，モデリン

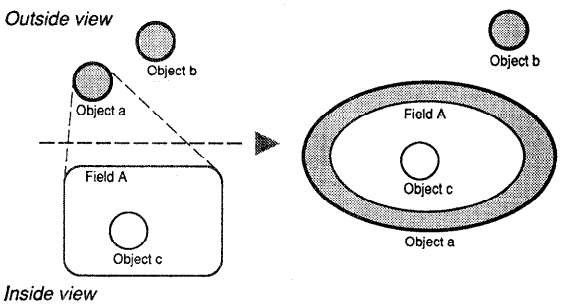


図 7 オブジェクトと場の重ね合わせ
Fig. 7 Field as object.

```
Entity subclass: #ObjA
  overlay: FieldA
  instanceVariableNames: 'i1 i2'
  classVariableNames: 'c1'
  poolDictionaries: 'p1' !
```

図 8 場に重ねられたオブジェクトの記述
Fig. 8 Description of object attended to field.

グの視点によるオブジェクトと場の排他的なとらえ方を実現していない．われわれはオブジェクトと場の重ね合わせによってこれらの視点を統合し，必要に応じてそれぞれの性質をとらえうる機構を実現した．

記述の形式は，すでに定義されている場に加えてオブジェクトとしての性質を記述することになり，オブジェクトの記述とほぼ同様の形式となる（**図8**）．

この機構によってモデル化の際の二つの視点，グループを構成するオブジェクトの視点，およびグループを外から一つのまとまりとしてとらえる視点をプログラム上においても分離し，記述することができる．オブジェクトのグループをオブジェクトの動作環境としてだけではなく，グループをより抽象度の高いオブジェクトとして定義することができ，オブジェクトの機能に応じた問題の階層的な表現を可能にするという利点がある．

3.4 通信形態

オブジェクトはメッセージを交換することによって互いに通信し合う．通信の形態は，(i)オブジェクトからオブジェクトへの直接通信，(ii)オブジェクトから不特定多数のオブジェクトへの場を介した間接通信に大きく分けられる．

(i) 直接通信

それぞれが並行に動作しているオブジェクト間での直接通信は，メッセージの送信方法によって四つの形態に分かれる[6]．

1. 同期送信…送信オブジェクトはメッセージが相手オブジェクトに受理されるのを確認した後，次の動作に移る．記述形式は次のとおりである．記述の中で，obj は送信先のオブジェクトの識別子を，message は送信メッセージを，arg はメッセージの引数をそれぞれ表す．

   obj<-message : arg.

2. 同期交信…送信オブジェクトはメッセージが相手オブジェクトに受理，実行され，結果を受けとるまで，次の動作を待つ．記述形式は次のとおりである．

   x :=obj<-message : arg.

3. 非同期送信…送信オブジェクトはメッセージを送

信後，すぐに次の動作に移る．記述形式は次のとおりである．&をつけることで非同期であることを示す．

obj<-message : arg &.

4. 非同期交信…送信オブジェクトはメッセージを送信後，すぐに次の動作に移る．相手オブジェクトでの実行結果が必要になった時点で，結果を待つ．記述形式は次のとおりである．

x :=obj<-message : arg &.

（ii） 間接通信

場に対してオブジェクトより送られたメッセージは，場のブロードキャスト機能によって場に存在する他の不特定多数のオブジェクトに送られる．間接通信による送信メッセージは場内の各オブジェクトに非同期的に送られる．

ブロードキャストによる間接通信は会議やグループ・コミュニケーションなどの協調動作におけるグループ内相互作用の表現をより容易にする．

メッセージはオブジェクトより次の形式で場に対して送られる．

field<<-message : arg.

4. 処理系の実装

Macintosh 上で動作する Smalltalk/V を用いて処理系を実装した．Smalltalk を基本とし，その機能を拡張する形で実装し，オブジェクトや場の機能，メッセージ通信の制御を実現するために，クラス階層中にいくつかのクラスを作成している．

（i） オブジェクト・場の実現

図 9 にオブジェクトと場の構成を示す．オブジェクトは二つのクラスのインスタンスから構成される．クラス Entity はすべてのオブジェクトのルートとなるクラスであり，オブジェクトは Entity のサブクラスとして定義される．状態変数，メソッドを持ち，オブジェクトそれぞれの具体的な振舞いをこの部分で実現する．クラス MessageHandler はオブジェクト間でのメッセージの送受に関わるさまざまな制御を行う．メッセージは必ずこの部分を経由して送受信される．具体的には，メッセージの同期処理や受信メッセージに対応したメソッドの起動などである．MessageHandler のインスタンスは，実行時には Smalltalk のプロセスとして存在している．したがって，疑似的ではあるがオブジェクトは一つの独立した並行プロセスとして考えることができる．これによって，並行に動作する処理実体をオブジェクトとしてとらえ，記述可能である．

場も同様に，クラス Field，クラス Broadcast のインスタンスから構成される．すべての場はクラス Field のサブクラスとして定義され，またクラス Broadcast は場に対して送られたメッセージをブロードキャストするための機能を実現している．

（ii） メッセージ通信の実現

3.4 節で述べた通信形態を Smalltalk で扱うことのできる構文へ変換するためにパーサを用意した．

メッセージ通信は**図 10** に示すように，MessageHandler を通して間接的に行われる．具体的なメッセージの送受の手順は次のとおりである．

1. 送信するメッセージはいったん自分の MessageHandler へ送られる．
2. MessageHandler では送られてきたメッセージに，通信形態，送信側の識別子の情報を付加し，これをメッセージ message : の引数として送信先の MessageHandler へ送る．受信側 MessageHandler はメソッド message の中でメッセージを自身のメッセージ・バッファへ入れる．通信形態が非同期型であれば，この時点で送信側は次の動作に移ることができる．通信形態が同期型であれば送信先からの返答（reply :）を待つ．
3. メッセージ・バッファ中のメッセージは，先頭から取り出され，メッセージ・セレクタに従ってメッセージを実行するための条件が調べられる．条件が満たされていなければ，バッファに戻し，

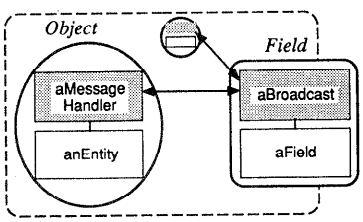


図 9 オブジェクトと場の実現
Fig. 9 Implementation of objects and fields.

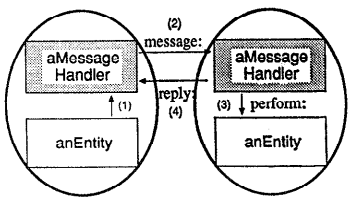


図 10 メッセージ通信の実現
Fig. 10 Implementation of communication method.

次のメッセージについて同様の処理を行う．実行可能であればメッセージを実行する．実行はメッセージ perform：を Entity に送信することによって，メッセージに対応した Entity 中のメソッドを起動して行われる．

4. 実行されたメッセージの通信形態が同期型であれば，メソッドの実行後，メッセージの送信元の MessageHandler に戻り値を引数として返答メッセージ reply：を送る．

このように，メッセージを扱う部分を独立させたことにより，メッセージにさまざまな処理を加えることが可能である[15]．

5. 応　　用

例題として，ここではスケジュールの割当ての問題を考える[14]．ある一つの部屋で会議Aと会議Bを開催することを考える．この場合，二つのレベルでのスケジュールの調整が行われなければならない．一つ目が，会議Aの参加者内での予定の調整，会議Bの参加者内での予定の調整である．もう一つが，一つの部屋をめぐる会議Aと会議Bの間での予定の調整である．

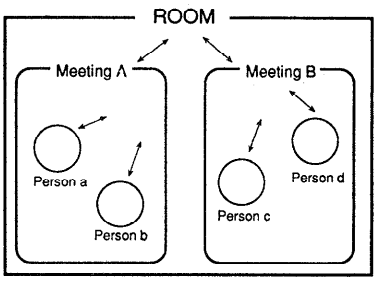


図 11 予定の割当て問題における階層構造
Fig. 11 Hierarchical structure for event scheduling problem.

```
Entity subclass: #Person
   instanceVariableNames: '
      meetingField   "参加する会議"
      n      "会議の参加者数"
      c      "候補日時を受け取った人数" '
   classVariableNames: ''
   poolDictionaries: '' !

! Person methods !
initialize
   ....!

meetingSchedule: aDays
"会議の開催予定日の中から都合のよい日時を選ぶ"
| aConvenientDays |
   aConvenientDays := self checkSchedule: aDays.
   meetingField <<- candidate: aConvenientDays!

candidate: aDays
"他の参加者からの候補日時を集め
 最終的な候補日時を決める"
      aCandidate add: aDays.
      c := c + 1.
      c = n ifTrue: [
         meetingField candidate: aCandidate] !

.....!!


Field subclass: #MeetingField
   instanceVariableNames: 'candidateDays decidedDay'
   classVariableNames: ''
   poolDictionaries: '' !

! MeetingField methods !
candidate: aDays
   candidateDays := aDays!

decidedDay
   ^decidedDay !!
```

(i) 参加者と会議

```
Entity subclass: #Meeting
   overlay: MeetingField
   instanceVariableNames: 'roomField candidateDays
         decidedDay n c'
   classVariableNames: ''
   poolDictionaries: '' !

! Meeting methods !
initialize
   ....!

notifyCandidate
"日時の候補を場を通して他の会議に知らせる"
   roomField <<- candidate: candidateDays of: myself!

candidate: aDays of: aMeeting
"他の会議の情報を集め日時を決定する"
   aCandidate add: aDays.
   c := c + 1.
   c = n ifTrue: [
      decidedDay := self makeDecision] !

.....!!

Field subclass: #Room
   instanceVariableNames: ''
   classVariableNames: ''
   poolDictionaries: '' !

.....!!
```

(ii) 会議と部屋

図 12 予定の割当て問題のプログラム例
Fig. 12 Program example for event scheduling problem.

この様子を**図 11** に示す．会議は，会議の参加者にとって場であり，参加者は会議という場の中でお互いに情報を交換しあって会議の開催日時の候補を決定していく．一方，A，Bそれぞれの会議は部屋の使用に関してお互いに情報を交換し合い，調整しなければならない．ここでは，会議は情報交換のため，メッセージを交換し合い，部屋という場の中で活動するオブジェクトである．

会議がもつオブジェクトと場の二つの側面は，われわれのモデルでは，オブジェクトと場という二つの要素によって別々に表現される．これにより，問題のもつ階層性を記述の段階においても，それぞれのモデル化の視点に合わせて表現することが可能となる．

プログラムの骨格は**図 12** のようになる．問題がもつ階層性を反映し，プログラム自身も二つのレベルから記述される．

（ⅰ） 参加者はオブジェクトとしてクラス Person で定義される．同一の会議への参加者はクラス Meeting-Field で定義される場に存在する．場 Meeting-Field 内では，会議開催予定の期間が場内にブロードキャストされ，これを受けとった参加者はその中から都合のよい日時の候補を選び，場内にブロードキャストする（メソッド meetingSchedule）．他の参加者からの候補日時を受けとると，それを考慮して最終的な候補日時を決め，場の状態変数 candidateDays に登録する（メソッド candidate）．

（ⅱ） 会議はオブジェクトとしてクラス Meeting で定義される．実際には，場を表す MeetingField にこれが重ね合わせられている．会議 Meeting は，参加者間の相互作用により候補日時が決定し，candidate-Days が決まると，場 Room にこれをブロードキャストして会議開催の日時を交換し，会議が部屋を使用する時間を決定し，decidedDay に記録する．変数 candidateDays, decidedDay は Meeting, Meeting-Field の間で共有され，二つのレベル間での情報が交換される．

Kamui-88[11] や組織計算モデル[12] における，場を間接通信の媒体としてとらえるアプローチでは，（ⅰ）（ⅱ）に対してそれぞれ Person と MeetingField, Meeting と Room というオブジェクトと場でその関係を表現できるが，会議がもつ二つの側面を統合して扱うことができない．この問題では，会議という実体は，参加者に対しての動作環境，場としての役割をもつが，同時に，より上位の部屋という場に対しては，他の会議と情報を交換し合うオブジェクトとして振舞う．したがって，参加者，会議，部屋の間にある関係を階層的に表現することにより，自然なモデリングが可能である．Kamui-88，組織計算モデルでは場の中にさらに場が存在することはなく，また，それがオブジェクトとして振舞うこともないので，二つのレベルにまたがって存在する会議のような実体をプログラム上に表現するためには，プログラマが意識して操作する必要がある．われわれのモデルでは，より自然な形式で容易にこの状況を表現することが可能となる．

6. おわりに

本論文では，個々の実体を表現するのに有効なオブジェクト指向モデルに基づき，これにオブジェクトのグループの表現を容易にするために場の概念を導入した Object-Field モデルを示した．グループによる協調動作の表現には，グループを構成するオブジェクトが不特定多数の相手と通信する必要があり，オブジェクトとは異なる構成要素として通信媒体となる場を新たに導入した．また，グループを異なる二つの視点からモデル化する必要があることから，オブジェクトと場の重ね合わせについても述べた．グループをオブジェクトとして扱うことによって複数のグループから構成されるグループのような，階層的なグループを表現することができる．このことは，多数のオブジェクトからなる大規模な問題のモデル化において，オブジェクトをその機能により分類し，階層化された複数のレベルから問題を表現できるという利点があり，重要なモデル化の視点である．そして，各実体を表すオブジェクトと，実体のグループを表現する場の二つの要素によるモデル化は，実世界における実体の活動をより自然かつ容易に表現しうる．

今後の課題として，場がオブジェクトに及ぼす間接的な影響について検討する必要がある．オブジェクトは自身の目的に従って場を自由に移動することができる．このとき，オブジェクトの振舞いは，オブジェクト自身が固有にもつ振舞いだけではなく，そのオブジェクトが存在する場に応じた何らかの変化，影響を受けた特別な振舞いをすると考えられる．たとえば，会議に出席しようとしている人（オブジェクト）を考えてみる．この人は，ある会議（場）には発表者として参加し，また別の会議には司会者として参加する．このように，参加する会議によってその人の会議での役割，性質は変化する．そのため，オブジェクトの性質

の変化として，メソッドおよびメッセージ・セレクタに影響を与える機構が必要である．

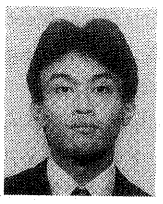
オブジェクトの振舞いの場への適応は，協調的な動作を行うために必要な条件の一つともいえ，これに対する検討は重要な課題である．

謝辞 本研究を行うにあたり貴重なご助言をしていただいた中京大学情報科学部 福村晃夫教授，名古屋大学工学部 稲垣康善教授，鳥脇純一郎教授に深謝するとともに，多くの討論をしていただいた研究室の皆様，ならびに杉野花津江氏に感謝いたします．

参考文献

1) Meyer, B.: *Object-oriented Software Construction*, p. 534, Prentice Hall (1988).
2) Goldberg, A. J. and Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley (1983).
3) Stroustrup, B.: *The C++ Programming Language*, Addison-Wesley (1986).
4) 石川 裕，所真理雄：オブジェクト指向並行プログラミング言語，情報処理，Vol. 29, No. 4, pp. 325-333 (1988).
5) Nelson, M. L.: Concurrency & Object-Oriented Programming, *ACM SIGPLAN Notices*, Vol. 26, No. 10, pp. 63-72 (1991).
6) Yonezawa, A., Briot, J. and Shibayama, E.: Object-Oriented Concurrent Programming in ABCL/1 :, *Proc. of OOPSLA '86*, pp. 258-268 (1986).
7) Tokoro, M. and Ishikawa, Y.: Concurrent Programming in Orient84/K: An Object-Oriented Knowledge Representation Language, *ACM SIGPLAN Notices*, Vol. 21, No. 10, pp. 39-48 (1986).
8) Blake, E. and Cook, S.: On Including Part Hierarchies in Object-Oriented Languages, with an Implementation in Smalltalk, *Proc. of ECOOP '87*, pp. 41-50 (1987).
9) Craske, N.: SNOOPS: An Object-oriented Language Enhancement Supporting Dynamic Program Reconfiguration, *ACM SIGPLAN Notices*, Vol. 26, No. 10, pp. 53-62 (1991).
10) Helm, R., Holland, I. M. and Gangopadhyay, D.: Contracts: Specifying Behavioral Compositions in Object-Oriented Systems, *Proc. of ECOOP/OOPSLA '90*, pp. 169-180 (1990).
11) 渡辺慎哉，原田康徳，三谷和史，宮本衛市：場とイベントによる並列計算モデル—Kamui 88，コンピュータソフトウェア，Vol. 6, No. 1, pp. 41-55 (1989).
12) 丸一威雄，市川正紀，所真理雄：自律的エージェントからなる組織計算モデルと分散協調問題解決への応用，情報処理学会論文誌，Vol. 31, No. 12, pp. 1768-1779 (1990).
13) 丸一威雄，所真理雄：分散型知識処理に適したオブジェクト指向ルールベースプログラミング，コンピュータソフトウェア，Vol. 5, No. 4, pp. 27-39 (1988).
14) Suzuki, H., Watanabe, T. and Sugie, N.: A Scheduling Facility for Personal Tasks on the Basis of Agents Concept, 第 43 回情報処理学会全国大会論文集，pp. 5/79-80 (1991).
15) Pascoe, G. A.: Encapsulators: A New Software Paradigm in Smalltalk-80, *Proc. of OOPSLA '86*, pp. 341-346 (1986).
16) Nishio, F., Watanabe, T. and Sugie, N.: Design and Implementation of a Programming Language Based on Objects and Fields, *Proc. of TENCON '92*, pp. 623-627 (1992).

(平成 4 年 10 月 20 日受付)
(平成 5 年 9 月 8 日採録)



西尾 郁彦（正会員）

1968 年生．1991 年名古屋大学工学部情報工学科卒業．1993 年同大学院博士前期課程修了．同年ソニー株式会社入社．手書き文字認識の研究・開発に従事．在学中，オブジェクト指向プログラミングに興味を有し，研究．

渡辺 豊英（正会員）

1948 年生．1972 年京都大学理学部修了．1974 年同大学院工学研究科数理工学専攻修士課程修了．1975 年同博士課程中途退学．同年京都大学大型計算機センター助手．1987 年名古屋大学工学部情報工学教室助教授．京都大学工学博士．統合化環境，分散協調環境，データベース環境，データベースの高度インタフェース，知的 CAI，文書理解，地図理解に興味を持つ．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，ACM，IEEE Computer Society，AAAI 各会員．

杉江 昇（正会員）

昭和 7 年生．昭和 32 年名古屋大学工学部電気工学科卒業．電子技術総合研究所バイオニクス研究室長，視覚情報研究室長を経て，現在名古屋大学工学部情報工学科教授．視覚情報処理，神経科学，自然言語などの教育・研究に従事．工学博士．