B6-3

オブジェクト指向分析・設計言語Hoopにおける分散協調型モデルの表現方法

An Expression of Distributed Cooperative Model in Object-Oriented Analysis and Design Language Hoop

小西 裕治† 中所 武司†
Yuji KONISHI Takeshi CHUSHO

†明治大学 情報科学科
Department of Computer Science, Meiji University

概要

効率良くアプリケーションソフトウェアを開発することを目的として、分析・設計段階で使用するオブジェクト指向言語Hoopを開発中である。本発表では、分散協調型モデルを記述する機能について報告する。Hoopでは、オブジェクト指向のメッセージパッシングの概念を拡張し、メッセージを組み合わせたメッセージセットによって、オブジェクト指向分析・設計段階での分散協調型モデルの表現に必要となる多様な機能を実現した。

1 はじめに

情報化社会の到来により、コンピュータの必要性が高まるとともに、アプリケーションソフトウェアの需要は延び続けると思われる。

また、コンピュータはネットワークにつながれているのが当然のことのようになってきており、それに伴いアプリケーションがネットワークを通して情報のやりとりをすることがあたりまえになってきている。

このようなアプリケーションソフトウェアを効率良く開発するための開発環境が必要とされている。

「ドメインモデル≡計算モデル」を志向したアプリケーションソフトウエア開発環境であるM-base[1]の一環として、アプリケーションソフトウェア開発用のオブジェクト指向言語であるHoopを開発している。

Hoopは、分析・設計段階で使用するための言語であり、以下の概念をもとに設計されている。

- 上流過程で使用し、「分析≡設計≡プログラミング」を目標とする。
- オブジェクト指向の基本概念を素直に表現できる。
- 自立的なオブジェクトの並行処理(concurrency)を前提とし、ネットワークを通してメッセージのやりとりが出来る。

本発表では、Hoopにおけるオブジェクト指向の概念をベースにした分散協調型モデルについて述べる。

2 設計思想

“一般に、ソフトウェア技術は「モデリング&シミュレーション」技術である。従来のプログラミングではシミュレーション可能なモデルを作るために論理を駆使してきたが、そのためには熟練技術が必要であった。...対象モデルを計算モデルに変換するという作業は残った。”[2]という観点から、モデルの表現能力がソフトウェアを作る際の重要な要因であると考えた。

Hoopでは、オブジェクト指向の特長のひとつで

ある強力なモデル化の能力を使い、対象世界を分散協調型モデルで表現する。

オブジェクトにメッセージを送ることによって計算を抽象化しようとする[3]オブジェクト指向プログラミング言語としては、ABCL/1 [5]、ConcurrentSmalltalk [4]などがある。これらの言語ではメッセージパッシングの機能によって豊かな表現力を得ている。

Hoopの分散協調モデルではオブジェクト指向のメッセージパッシングの概念を拡張しメッセージを組み合わせることによって、オブジェクト指向分析・設計段階での分散協調型モデルの表現に必要となる多様な機能を実現した。

3 モデルの概要

オブジェクト指向における分散協調型モデルは、“複数個の自立的機能を持つオブジェクトがお互いにメッセージをやり取りしながら協調して問題解決にあたる”[2]ものである。

Hoopのモデルでは、オブジェクトどうしが直接メッセージをやりとりするのではなく、“宛先のオブジェクト”+“メッセージ”をひと組みとし、これをいくつか組み合わせたものをメッセージセットとし、メッセージセットをオブジェクトどうしがやりとりする。メッセージセットにより、逐次処理、同期処理などが表現できる。

また、オブジェクトについては、それぞれの役割によって3種類用意した。

4 オブジェクト

ワークステーションなどのネットワークに接続された環境では、個々のオブジェクトが単一のコンピュータ上ではなく、ネットワーク上に分散されている場合があるため、オブジェクトがネットワークのノードに対応する必要がある。

しかし、全てのオブジェクトがネットワークのノードに対応するような構成は現実的ではない。

また、ネットワーク上のノードに対応するオブジェクトは並列に動くことが出来るが、単一のコンピュータ上のオブジェクトについても並列に動くことが望ましい。ただし全てのオブジェクトが並列に動く必要性はない。

以上のようなことを実現するために以下の3種類のオブジェクトを定義する(図1)。

ノードオブジェクト ネットワーク上のノードに対応するオブジェクトであり、内部にプロセスオブジェクトを持つことができる。

プロセスオブジェクト プロセスオブジェクトひとつに、仮想的にプロセッサが割り当てられていると考える。他のプロセスオブジェクトとは並列に動くことができる。内部にサブオブジェクトを持つことができる。内部のサブオブジェクトとは同期しながら動く。

サブオブジェクト 同一のプロセスオブジェクトの内部にあるサブオブジェクトどうしは、並列には動作できない。内部にサブオブジェクトを持つことができる。

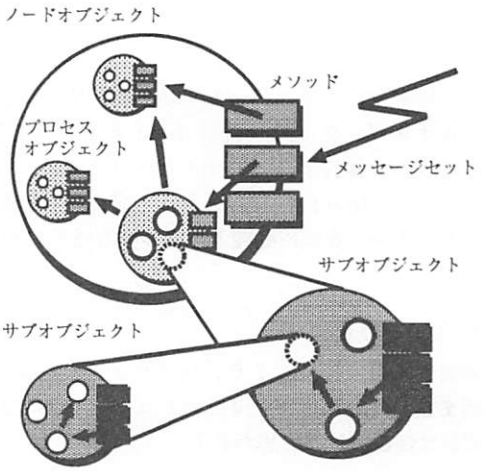

図1 オブジェクト

5 メッセージセット

5.1 メッセージセットの表現

メッセージセットはメッセージを組み合わせることによって構成する。メッセージセット M は以下のように与えられる。

$$M ::= M_s \| M_p \| X$$
$$M_s ::= \{M_1, M_2, \ldots, M_n\}$$
$$M_p ::= \{M_1 | M_2 | \ldots | M_n\}$$
$$X ::= (o, m)$$

o はオブジェクト、m はメッセージ

$\alpha \| \beta$ は α、β のどちらか

M_sが直列のメッセージセットであり、M_pが並列のメッセージセットである。$X ::= (o,m)$は、オブジェクトoにメッセージmを送るという意味である。M_s、M_pを組み合わせてメッセージセットMを作る。以下に説明をする。

5.2 直列メッセージセットの基本形

アクターモデル[6]のメッセージは

[request: <a-request> reply-to: <a-continuation>]

のように表せる。"request: <a-request>"は要求するオペレーションおよび必要なデータであり、"reply-to: <a-continuation>"は結果の送り先になる。操作や計算の結果を送る時には

[reply: <a-result>]

というようなメッセージを使うことがある。つまり、メッセージを送る際に必要となる主な要素としては

{ 宛先のオブジェクト, メッセージ,
結果の送り先のオブジェクト }

である。

Hoopでは、"結果の送り先に対するメッセージ"も一緒に送る。これは

$\{(o_1,m_1),(o_2,m_2)\}$
o_1は、宛先のオブジェクト
m_1は、宛先のオブジェクトに対するメッセージ
o_2は、結果の送り先のオブジェクト
m_2は、結果の送り先に対するメッセージ

と表すことが出来る。このようにオブジェクトとメッセージの組み(o,m)を複数組み合わせたものが、Hoopの直列のメッセージセットの基本的な考え方である。

直列にメッセージを送るのは、以下のような場合である。

例1) ある書類oを提出する時に、Aさん、Bさん、Cさんにハンコをもらわなければならず、かつAさん、Bさん、Cさんの順番でハンコをもらわなければならない。

例2) Aさんは書類o1をもらうと、書類o2を作り、Bさんに送る。Bさんは書類o2をもらうと、書類o3を作り、Cさんに送る。

例1の場合は共通の書類oが順番にまわって行き、例2の場合は次々に書類を作って行く。

直列のメッセージセットは以下のように与えられる。

$$M ::= \{M_1, M_2, \ldots, M_n\}\ (n \geq 1)$$

簡単な例としては

$$\{(o_1,m_1),(o_2,m_2),\ldots,(o_n,m_n)\}$$

である。

オブジェクトo_iはメッセージm_iを受け取るとオブジェクトo_{i+1}に対して

$$\{(o_{i+1},m_{i+1}),(o_{i+2},m_{i+2}),\ldots,(o_n,m_n)\}$$

を送る。

5.3 並列メッセージセットの基本形

オブジェクトoがオブジェクトo_1にメッセージm_1、オブジェクトo_2にメッセージm_2、...、オブジェクトo_nにメッセージm_nを送りたいとする。しかも、メッセージの送られる順番は問わない時は一度に送れれば便利である。

並列にメッセージを送るのは、以下のような場合である。

例3) 会議を開催する時に、Aさん、Bさん、Cさんに出欠の問い合わせをする。ただし、Aさん、Bさん、Cさんのどの順番で返事がかえってきても良い。

例4) Aさんには書類o1の作成、Bさんには書類o2の作成、Cさんには書類o3の作成を頼み、三つの書類ができた時点で書類をまとめる。

並列のメッセージセットは以下のように与えられる。

$$M ::= \{M_1|M_2|\ldots|M_n\}\ (n \geq 1)$$

簡単な例としては

$$\{(o_1,m_1)|(o_2,m_2)|\ldots|(o_n,m_n)\}$$

である。あるオブジェクトが上記のメッセージセットを送ると、オブジェクト$o_1,o_2,\ldots,o_n$にそれぞれメッセージ$m_1,m_2,\ldots,m_n$が送られる。

6 分散協調型モデルの例

Hoopにおける分散協調型モデルの例として"会議開催事務処理"[2]を示す。

6.1 会議開催事務処理

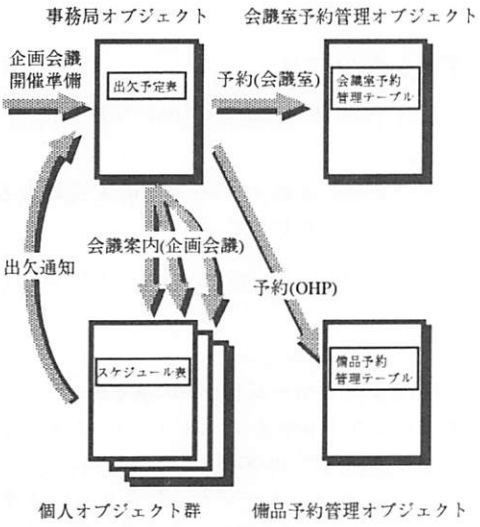

図 2 会議開催事務処理

“会議開催事務処理”は以下の 4 種類のオブジェクトからなる (図 2)。

事務局オブジェクト 会議開催の要求メッセージを受けると、会議室の予約、OHP の予約、その会議のメンバへの会議開催案内などのためのメッセージを各々の担当オブジェクトへ送り、その後、メンバからの出欠通知のメッセージを受け取り、管理する。

会議室予約管理オブジェクト 会議室予約の要求メッセージを受け取り、予約管理する。

備品管理オブジェクト 備品予約の要求メッセージを受け取り、予約管理する。

個人オブジェクト 会議開催案内のメッセージを受け取り、出欠通知のメッセージを送る。

6.2 モデル化

これらのオブジェクトは、ネットワーク上の別のノードにあっても良いものなので、ノードオブジェクトとする。ただし、これらのオブジェクトは稼働時に同じノード上にあっても同じ機能をはたす。

事務局オブジェクトが企画会議開催準備のメッセージを受けた時に、各個人オブジェクトに送るメッセージセットの例を示す。

$$
\begin{aligned}
\{\ & \\
& \{\ (\text{個人}_1, \text{会議案内}), (\text{事務局}, \text{出欠通知})\ \} \mid \\
& \{\ (\text{個人}_2, \text{会議案内}), (\text{事務局}, \text{出欠通知})\ \} \mid \\
& \quad \vdots \qquad\qquad \vdots \\
& \{\ (\text{個人}_n, \text{会議案内}), (\text{事務局}, \text{出欠通知})\ \} \\
\}\ &
\end{aligned}
$$

個人オブジェクトに対して、会議案内を送り、返事として出欠通知をもらう。

個人に対する会議案内メッセージは到着の順番は関係ないので並列のメッセージセットとして送れる。

ノードオブジェクトとメッセージセットを使うことによって、シンプルにモデル化が出来る。

7 おわりに

アクターモデルのメッセージを拡張し、オブジェクトとメッセージの組みを直列、並列に合成したものであるメッセージセットを導入することによって逐次処理、同期処理などが表現できた。

これらの機能はネットワークを使用したアプリケーションソフトウェアを作る際に多いに役にたつと思われる。

今後、Hoop を完成させるために

- 直列と並列のメッセージセットを組み合わせた場合の処理。
- メッセージセットの引数の指定方法。
- メッセージセットの delegation。

などを検討していく。

参考文献

[1] 中所 武司: M-base:「ドメインモデル≡計算モデル」を志向したアプリケーションソフトウエア開発環境の基本概念, 情報処理学会 ソフトウェア工学研究会資料 95-SE-104-4, 1995

[2] 中所 武司: ソフトウェア危機とプログラミングパラダイム“わかりやすさ”の追求, 啓学出版, 1992.

[3] 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, 1988, pp.325-333.

[4] 横手 靖彦, 所 真理雄: 並行オブジェクト指向言語 ConcurrentSmalltalk, コンピュータソフトウェア, Vol.2, No.4, 1985, pp.582-598.

[5] 米澤 明憲, 柴山 悦哉, J.-P.Briot, 本田 康晃, 高田 敏弘: オブジェクト指向に基づく並列情報処理モデル ABCM/1 とその記述言語 ABCL/1, コンピュータソフトウェア, Vol.3 No.3, 1986, pp.9-23.

[6] 米澤 明憲: ACTOR 理論について, 情報処理, Vol.20 No.7, 1979, pp.581-589.