題目　エンドユーザコンピューティングのための分散オブジェクト指向設計技法

英文題目　Distributed object-oriented design for end-user computing

研究者名　　中所　武司
所属　　明治大学 理工学部 情報科学科
ソフトウェア工学研究室

概要

近年，ワークステーションやパソコンの普及およびそれらをつなぐネットワークの普及と共に，業務の専門家が自ら情報システムを構築する必要性が高まっている．これに対応するため，業務の専門家が，自ら業務モデルを構築することがそのまま情報システムの構築につながるような技術として，分散オブジェクト指向設計技法を開発している．

このようなエンドユーザコンピューティングの促進のためには，プログラミングの概念を排した新しいパラダイムが必要であるという観点から，

・A domain model ≡ a computation model （業務モデルと計算モデルの一致）
・Analysis ≡ design ≡ programming （分析，設計，プログラミングの一体化）

という基本コンセプトを設定し，その実現の第１段階として，ワークフローシステムやグループウェアに代表される分散オフィスシステムのモデル化をマクロレベルとミクロレベルの２階層に分けて行う方式を具体化している．

マクロレベルではオブジェクト指向概念に基づく分散協調型問題解決モデルを用いて業務モデルの動的ふるまいを表現できるようなモデリングツールとそこで用いるスクリプト言語の基本機能を規定した．モデリングツールはアイコン操作を基本としたビジュアルな機能を実現した．

スクリプト言語の設計においては，オブジェクトをネットワーク上のノード割り当て方法と並行処理の可否によって３種類に分類すると共に，大規模化への対応としてオブジェクトのネスト構造を導入した．オブジェクト間の交信は，従来の１つのメッセージではなく，メッセージセット単位で行う方式にして協調作業のための業務の委託方法に柔軟性を持たせた．このメッセージセット指定機能により，本来メッセージの流れを管理する立場のオブジェクトがそれを指定することができるようになり，従来のワークフローシステムにみられるような全体のメッセージフローを管理するメタなシステムやオブジェクトを導入する必要がなくなった．

今後は，これらの技術をモデリング＆シミュレーション機能として実現するツールを開発する．特に特定応用分野向きコンポーネントウェアとの連携による分散協調型システム構築技法を実現するために，言語処理系，シミュレータ，ブラウザなどを開発を行い，具体的なアプリケーションの開発に適用し，実用性を評価する．

成果の要約（指定項目に関して）

■全体の目標と今年度の達成度

・第１ページの「概要」に記載.

■用途

・次ページ以降の本文中，第２章に記載.

■従来技術との比較

・次ページ以降の本文中，第３章３．１および第５章５．１，５．２に記載.

■今後の課題

・モデリング&シミュレーション機能を実現するツールを開発.
・特定のアプリケーションの開発に適用し，実用性を評価する.
・その評価改良と実用化のための機能拡張.

■有形成果物（論文を除く）

・下記ドキュメント
・モデリングツールのプロトタイププログラム

■デモンストレーションの可否

・上記モデリングツール（ただし，モックアップのレベル）

■実用化シナリオ

・平成８年度の基礎研究でプロトタイププログラムの開発とfeasibility studyを終了後，育成研究を申請し，実用化研究を推進する.

■ドキュメント

・中所：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ェ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, 104-4, pp.25-32（May.1995）
・中所：オブジェクト指向技術へのパラダイムシフト- Why > What > How -、情報処理学会オブジェクト指向'95 シンポジウム論文集、pp.327-329（Jun.1995）
・Takeshi CHUSHO, et al. : wwHww : An Application Framework for End-User Computing in Multi-organizational Office Network Systems、The 12th annual conf. of Japan Society for Software Science and Technology, pp.281-285 (Sep. 1995)
・小西，中所：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，ソフトウェア科学会第１２回大会論文集，pp.197-200, 1995.9.
・井上，中所：オブジェクト指向分析設計プロセスにおける創造的作業支援ツール，ソフトウェア科学会第１２回大会論文集，pp.161-164, 1995.9.
・松本，安斎，中所：wwHww：分散協調型アプリケーションフレームワークー電子帳票の分散処理方式ー，情報処理学会第５２回大会講演論文集（１），287-288 (Mar. 1996)
・安斎，松本，中所：wwHww：分散協調型アプリケーションフレームワークー応用システムへの適用ー，情報処理学会第５２回大会講演論文集（１），289-290 (Mar. 1996)

1 はじめに

近年，ワークステーションやパソコンの普及およびそれらをつなぐネットワークの普及と共に，業務の専門家が自ら情報システムを構築する必要性が高まっている．この傾向は多様なネットワークのさらなる普及により，情報のグローバル化と情報のパーソナル化という両面から一段と加速されるものと思われる．

このような分散コンピューティング，エンドユーザコンピューティング [12] という新しい動向に対応して，分散環境下でのオフィス業務（非定型業務）を業務の専門家自身がコンピュータ化するための設計技法を開発する必要がある．

本研究では，業務の専門家が自ら作り，自ら使うようなエンドユーザコンピューティングの促進のためには，プログラミングの概念を排した新しいソフトウェアパラダイムが必要であるという認識から，このような”作りやすさ”と”使いやすさ”を両立させるための基本的コンセプトとして，

- A domain model ≡ a computation model
  （業務モデルと計算モデルの一致）
- Analysis ≡ design ≡ programming
  （分析，設計，プログラミングの一体化）

を設定した．オブジェクト指向概念の分散協調型問題解決モデルに基づく”モデリング&シミュレーション”によってこれらのコンセプトを実現する分散オブジェクト指向設計技法を確立し，それに基づくアプリケーション開発環境 M-base[7] を開発中である．

マクロレベルではオブジェクト指向概念に基づく分散協調型問題解決モデルを用いて業務モデルの動的ふるまいを表現できるようなモデリングツールとそこで用いるスクリプト言語の基本機能を規定した．モデリングツールはアイコン操作を基本としたビジュアルな機能を実現した．

スクリプト言語の設計においては，オブジェクトをネットワーク上のノード割り当て方法と並行処理の可否によって３種類に分類すると共に，大規模化への対応としてオブジェクトのネスト構造を導入した．オブジェクト間の交信は，従来の１つのメッセージではなく，メッセージセット単位で行う方式にして協調作業のための業務の委託方法に柔軟性を持たせた．このメッセージセット指定機能により，本来メッセージの流れを管理する立場のオブジェクトがそれを指定することができるようになり，従来のワークフローシステムにみられるような全体のメッセージフローを管理するメタなシステムやオブジェクトを導入する必要がなくなった．

本報告では，２章では本研究の目的と対象，３章では本研究で基本とするモデリングプロセスの概要，４章ではモデリング&シミュレーション、５章では本報告の中心課題であるスクリプト言語の特徴について述べる．

2 研究の目的と対象

2.1 エンドユーザ

一般に，エンドユーザとして少なくとも以下の３種類が考えられる．

（ａ）基幹業務担当者

例えば，銀行のようなユーザ企業において，システム部門に対するエンドユーザ部門に所属する人達で，利用するソフトウエアはシステム部門が開発し，提供してくれる．

（ｂ）業務の専門家

一般にオフィスワーカーといわれるような人達で，DB検索や表計算などに市販のアプリケーションパッケージを利用する．

（ｃ）一般ユーザ

例えば，日常生活の中で銀行のＡＴＭを利用するような一般の人達で，将来，マルチメディア時代の主要ユーザとなる。

本研究では，（ａ）と（ｂ）のエンドユーザに対象を絞るが，特に非定型業務のコンピュータ化という視点では今後急速に増大すると思われる（ｂ）の方により重点を置く．

2.2 対象ソフトウエア

ソフトウエア開発の問題領域は多様であり，汎用的な設計プロセスは難しい．例えば，ビジネス分野の基幹業務向きのデータモデル中心の設計法，エンジニアリング分野の状態遷移モデル中心の設計法などのように，設計法が問題領域に依存するのは止むをえない．

本研究では，「すべての日常的な業務をコンピュータ化する」，即ち，「日常的業務はマニュアル化でき，マニュアル化できればコンピュータ化できる」という CS-life（Computer-supported Life：コンピュータ支援による豊かな生活）[9] の実現の観点から，上記の（ｂ）のエンドユーザが主体であり，オフィスでの非定型業務用アプリケーションが中心となるので，分散協調型モデルに基づく設計法を提案する．従って，CSCW (Computer-Supported cooperative Work) のツールやワークフロー

システム、さらにエージェントシステムも対象となる．規模的には，中，小規模のアプリケーションソフトウエアを想定することになるが，ネットワーク接続するによりシステムとしては大規模化することもある．

2.3 開発・保守形態

本研究では，基本的コンセプトとして，

「ドメインモデル≡計算モデル」

「分析≡設計≡プログラミング」

をかかげた．これは，問題領域の分析によりドメインモデル [6] を構築した時点でソフトウエアの開発を完了させようというものである．即ち，

「ソフト開発＝モデリング＋シミュレーション」

という図式で表現されるように，問題領域のモデルを作成し，そのモデル上でのシミュレーションによりモデルの妥当性を検証した後，実用に際しては，そのモデルをインタプリタにより実行するか，あるいは必要に応じて実際のプログラムを自動生成するという方法である．

このように最終的にはエンドユーザが自らの業務のアプリケーションソフトウエアを自ら開発し，自ら利用すること（applications of the end-users, by the end-users, for the end-users）を目標とするが，その実現に向けての技術課題は多い．そこで，研究のマイルストーンとして，エンドユーザが主体でシステムエンジニアの助けを借りて開発するが，保守はエンドユーザだけで行うレベルを設定する．

また，既存の類似システムが存在する場合は，クラスの設計やクラス間の関係の定義を開発の初期段階で行うことが可能であるが，前例のない新規開発の場合は、トップダウンに設計する必要がある．ただし、特定分野向きのコンポーネントウェアを導入して、プログラミングの複雑さを回避する。

さらに，大規模ソフトの場合は多人数開発になるので，各工程で仕様を検証しながら開発を進めるフェーズドアプローチをとるが，本研究では，エンドユーザコンピューティングの分野を対象とするため，まず核になる部分を試作し，それを改良しながら実用システムに仕立てあげるプロトタイプアプローチを想定する．

2.4 M-baseの研究項目

M-baseでは、エンドユーザ自身の手による、比較的小規模なアプリケーションの開発を支援する統合的な環境を実現するために、本報告での主要テーマであるスクリプト言語の研究を含め、以下の分野の研究が行われている。

- モデリングツール [17]
- ビジュアルなシミュレータ（アニメータ）
- スクリプト言語 [4]
- コンポーネントウェア [15]
- ユーザインタフェースの設計

3 モデリングプロセスの概要

3.1 2階層モデル

従来のオブジェクト指向設計技法は，設計プロセスの初期の段階でオブジェクトおよびオブジェクト間関係を定義するものが多く，オブジェクト間関係のビジュアルな表記法が豊富に導入されている．

これらのボトムアップ的アプローチは，既に開発運用実績を有する従来の基幹データベースを中心としたビジネスシステムの再構築のような場合には適用可能である．しかし，システム全体のマクロレベルの動的ふるまいの設計法があいまいであるため，今後増加していくと思われるエンドユーザコンピューティングに近い非定形業務の新規開発には適さない．

このような分野では，何をオブジェクトにするかを決定するためには，モデリングの初期の段階で，従来の技法であまり明確に示されていないオブジェクト群の動的なふるまいを記述することが重要である．本報告では，図1に示すような2階層モデルにより、基本的コンセプトを以下のように規定する。

（1）マクロモデルは,「ドメインモデル≡計算モデル」を実現するレベルであり，オブジェクト指向の分散協調型モデルとして位置づけられる．

（2）ミクロモデルは,「分析≡設計≡プログラミング」を実現するレベルであり，オブジェクト指向のクラス定義として位置づけられる．

3.2 ドメインモデルの構築手順

本研究では，オブジェクト指向の概念の発生学的定義 [10] に基づいてモデリングプロセスの形式化を行う．概要を図2に示す．詳細は文献 [7] に詳しい。

分析フェーズで構築されるオブジェクトベースのドメインモデルを以下のように OAM (Object-base Analysis Model) として表現する．

OAM = { O, M, T }

O = o[i]

M = m[i,j,n]

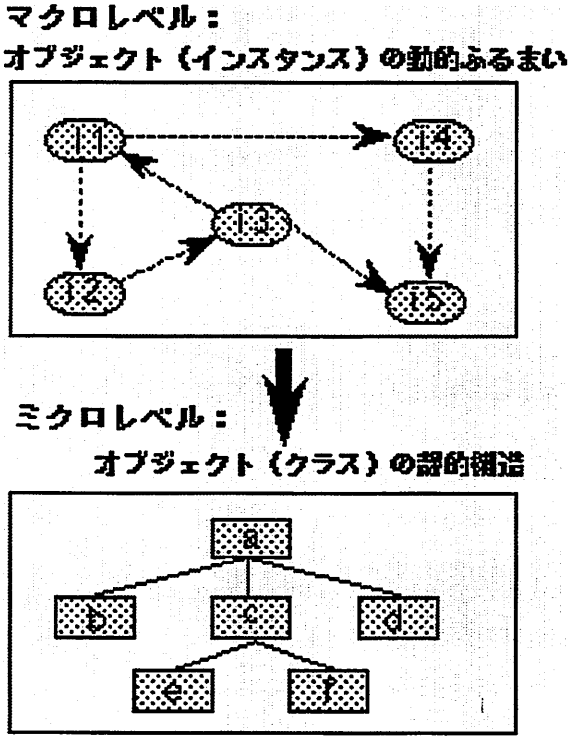

図 1: 2階層モデル

$$T = t[r]$$

ドメインモデルOAMは，オブジェクトの集合O，メッセージの集合M，メッセージ変換の集合Tの3つ組で規定する．o[i]は，ドメインモデルに含まれるi番目のオブジェクトである．m[i,j,n]は，i番目のオブジェクトo[i]からj番目のオブジェクトo[j]へ送信されるn番目の種類のメッセージである．メッセージ変換の集合Tは，あるオブジェクトo[j]があるメッセージm[i,j,n]を受信した後，メッセージの列 m[j,k1,n1], m[j,k2,n2], ... を送信するという関係を

t[r] : m[i,j,n] → m[j,k1,n1], m[j,k2,n2], ...

と表現したメッセージ変換の集合である。

このドメインモデルは，2階層モデルの下位の層に対応する設計モデルに詳細化される．設計モデルはクラスに基づいて構築されるので，以下のように CDM (Class-based Design Model) として表現する．

CDM = {MD, C, H}

設計モデルCDMは，メソッドの集合MD，クラスの集合C，クラスの階層関係の集合Hの3つ組で規定する．

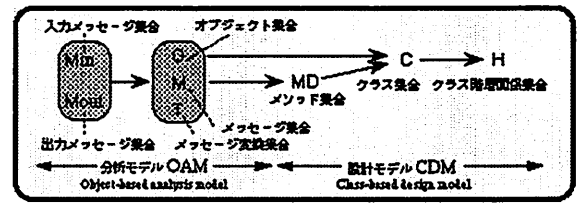

図 2: M-base のモデリングプロセス

4 モデリング&シミュレーション

4.1 メタファーベースのモデリング

オフィスの日常的業務（ルーチンワーク）は，メッセージ駆動型の分散協調型モデルを用いて次のように表現できる．

（1）ひとまとまりの日常的業務が割り当てられる一人または複数の人からなるグループを一つのオブジェクトに対応させる．

（2）一人又はグループの間で相互の通信手段として用いられている書類，メモ，電話，郵便，口頭連絡などはすべてメッセージとみなす．

（3）日常的業務における協調作業はメッセージの送受信によって遂行される．

本研究では，一つの業務を擬人化したオブジェクトに割り当てることにより，メタファーベースのモデル化を行う．実世界と同じように一つのオブジェクトに複数の業務を割り当てることも可能であるが，柔軟性と保守性を重視して「1オブジェクト1業務」の割り当てを原則とした．

4.2 モデリング&シミュレーション

M-baseでのシステム開発は、図3のような流れになる。「モデリング」とは、OAM、CDMのモデリングのことである。

エンドユーザはまずOAM段階として、オブジェクトとメッセージによって、システムの動的モデルをコンピュータ上でモデリングする。それからユーザは、モデリングしたシステムをシミュレートする。おかしな所があれば、すぐにモデリング画面に戻って原因を究明することができる。

このようにユーザが、オブジェクトのモデリングとシミュレーションを繰り返しながら、目的システムを組み上げてゆく「モデリング&シミュレーション」が、M-baseの開発環境の基本理念である。

上記はOAMについての説明だが、メソッド定義も

シミュレーションの動作に反映されるので、CDMでメソッドを定義した後でも、シミュレーションを行なってもよい。

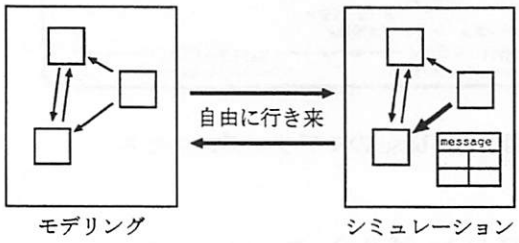

図 3: モデリング&シミュレーション

4.3 モデリングツール

モデリングツールは、主にマウス操作により動的モデルを構築するツールである。動的モデルは、オブジェクトとメッセージから成り、メッセージパッシングで動作する。

モデリングツールは、システムの動的モデルをアイコン表示などで視覚的に判りやすく表し、かつ、オブジェクト指向の概念を素直に表現することを目標としている。図4がモデリングツールのプロトタイプの画面である。

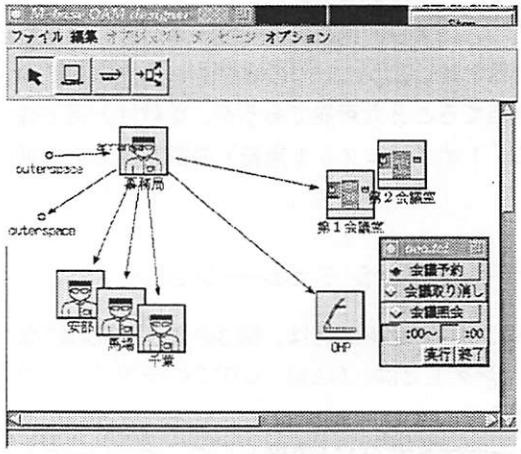

図 4: 開発中のモデリングツール

以下でモデリング時に使う画面上のパーツについて説明する。これらはシミュレーション画面でも使われる。

4.3.1 オブジェクトの表現

オブジェクトの表示方法は、図5のように幾つかあり、ユーザの好みに応じて切り替えることが出来る。

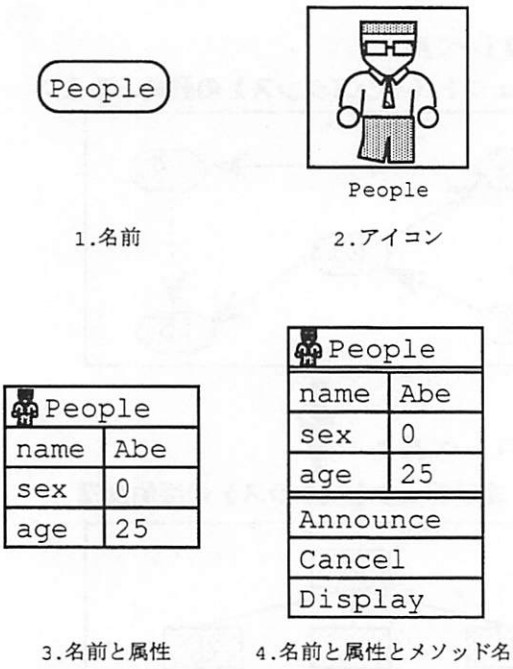

図 5: さまざまなオブジェクトの表示形態

4.3.2 メッセージの表現

図6がメッセージである。図4は開発中の画面なので、表示が少し違う。

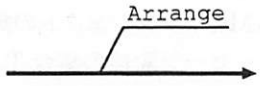

図 6: メッセージ

4.3.3 オブジェクトのグループ化

オブジェクトとメッセージを組み合わせてできたシステムを、1オブジェクトととらえ、他のシステムのモデリングに利用できる。一度に複数のシステムを画面上に並べたり、入れ子状態にしたりして編集できる。(図7)

4.3.4 システム外メタファ

システムの中に入らない、実世界のものに対するメタファを、アイコン等で、表示する。Fusion 法 [21] で言うところの「エージェント」に相当する。(図7下)

5 スクリプト言語 Hoop

グループウエアやワークフローシステムなどの分散協調型のアプリケーションソフトウェアを効率良く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

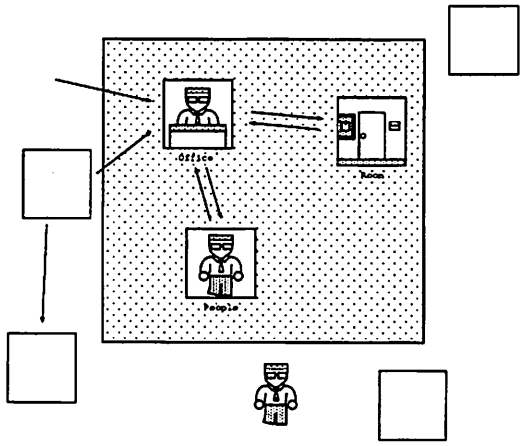

図 7: グループ化

ジェクト指向言語 **Hoop**[4] を開発中である。言語の基本機能として、オブジェクトをネットワーク上のノード割り当て方法と並行処理の可否によって 3 種類に分類すると共に、大規模化への対応としてオブジェクトのネスト構造を導入した。オブジェクト間の交信は、従来のひとつのメッセージではなく、メッセージセット単位でおこなう方式にして協調作業のための業務の委託方法に柔軟性を持たせた。さらに、分散協調型モデルの構築を形式化された手法でおこなうための手順を提示すると共に、グループウエアの代表的なアプリケーションパッケージであるスケジューリングシステムの一種を **Hoop** で記述し、記述の容易性、拡張性を確認した。

5.1 設計思想

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。

- 上流工程で使用し、「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。

オブジェクト指向の概念としては、分散協調型モデル、データ抽象化、クラスからのインスタンス生成、クラス階層と継承などがあるが、モデリング重視の立場から、特に分散協調型モデルを重視する。

“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。”[11] 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。

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 [2] オブジェクト指向プログラミング言語としては、ABCL/1[5]、Concurrent Smalltalk[16] 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、グループウエアやワークフローなどに必要な多様な機能を実現した。分散協調型アプリケーションの開発を目標とした
Hoop のおもな特徴は、以下のようなものである。

- オブジェクト間のメッセージの流れを記述するメッセージセット。
- ネットワーク上でのオブジェクトの柔軟な割り付けを可能とするオブジェクトのネスト構造。
- 分散協調型モデルの構築の手順を示す形式化された分析・設計手法。

5.2 Hoop の分散協調型モデル

5.2.1 メッセージセット

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”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て、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。すなわちメッセージセットとは、オブジェクト間に伝わるメッセージの流れを記述するためのものである。

メッセージセットを導入することにより、本来メッセージの流れを管理する立場のオブジェクトが、これを指定できるようになる。例えば、ワークフローシステムにおいて全体のワークフローを管理するメタなシステムあるいはオブジェクトが不要になり、純粋な分散協調型モデルを構築できる。

このメッセージセットは、先に述べた OAM モデルにおけるメッセージ変換集合 M の各要素を時系列的に複数個まとめたものに対応する。

5.2.2 メッセージセットの構文

$M ::= M' \| [cond] M'$
$M' ::= M_s \| M_p \| X$
$M_s ::= \{M_1, M_2, \ldots, M_n\}$
$M_p ::= \{M_1 | M_2 | \ldots | M_n\}$
$X ::= (obj, msg, arg_1, arg_2, \ldots, arg_n)\ (n \geq 0)$
objはオブジェクト
msgはメッセージ
$cond$は条件
arg_nは引数
$\alpha \| \beta$はα、βのどちらか

メッセージセットの基本単位は、

(obj, msg)

であり、引数がある場合は

$(obj, msg, arg_1, arg_2, \ldots, arg_n)$

となる。

5.2.3 直列のメッセージセット

メッセージの流れを表現するための従来の方法のひとつとして、アクターモデル [18] のメッセージでは

{ 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 }

のように表現する。

Hoop では、さらに "結果の送り先に対するメッセージ" も一緒に送る。これは

$\{(obj_1, msg_1), (obj_2, msg_2)\}$
obj_1は、宛先のオブジェクト
msg_1は、宛先のオブジェクトに対するメッセージ
obj_2は、結果の送り先のオブジェクト
msg_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 (obj, msg) 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。

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

簡単な例としては

$\{(obj_1, msg_1), (obj_2, msg_2), \ldots, (obj_n, msg_n)\}$

である。

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

$\{(obj_{i+1}, msg_{i+1}), (obj_{i+2}, msg_{i+2}), \ldots, (obj_n, msg_n)\}$

を送る。

5.2.4 並列のメッセージセット

オブジェクト o がオブジェクト obj_1にメッセージ msg_1、オブジェクト obj_2にメッセージ msg_2、...、オブジェクト obj_nにメッセージ msg_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。

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

簡単な例としては

$\{(obj_1, msg_1) | (obj_2, msg_2) | \ldots | (obj_n, msg_n)\}$

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 $obj_1, obj_2, \ldots, obj_n$にそれぞれメッセージ $msg_1, msg_2, \ldots, msg_n$が送られる。

5.3 モデル化の手順

モデル化の手順を明確にするために、文献 [5] でオブジェクト指向モデル化が試みられている "在庫管理問題" を取り上げて、具体的に検討し、以下のようなモデル化の手順でよいことを確認した。この手順の形式化の詳細は文献 [7] に詳しい。

(1) オブジェクトの抽出

メッセージのやり取りをするものをオブジェクトとする。

(2) 受付オブジェクトの抽出

抽出したオブジェクトの中から、外部とのメッセージのやり取りをするものを、受付オブジェクトにする。

(3) 受付オブジェクトが行う依頼の抽出

外部からのメッセージを受け取った後、他のオブジェクトに対して行う依頼を取り出す。"obj_1に対して msg_1 を頼んだ後、obj_2に対して msg_2を頼み、..." のような記述である。

(4) 依頼からメッセージセットへの変換

"obj_1に対して msg_1を頼んだ後、obj_2に対して msg_2 を頼み、..." という依頼は以下のようにメッセージセットに変換される。

$$\{ (obj_1, msg_1), (obj_2, msg_2), \ldots \}$$

(5) 個々のメッセージの作成

$msg_1, msg_2, \ldots$ を実装する。

5.4 例題への適用

モデル化の手順を、本システムが主要なターゲットとしているエンドユーザコンピューティングの分野の例題に当てはめ、有用性を確認する。例題として、会議開催事務処理問題 [11] を取り上げる。

この例題は、グループウエアの代表的なアプリケーションパッケージであるスケジューリングシステムの一種であると考えて良い。現在、例えばオフィスで、事務所に会議開催の指示が出されると、事務局はその会議のために会議室の予約と OHP などの備品の予約を行うと共に、会議開催通知を出席者に送り、出欠の返事を返してもらうという作業があったとしてこの業務を自動化するアプリケーションの開発を想定する。

5.4.1 オブジェクトの抽出

"会議開催事務処理問題" は以下の 4 種類のオブジェクトからなる (図 8)。

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。

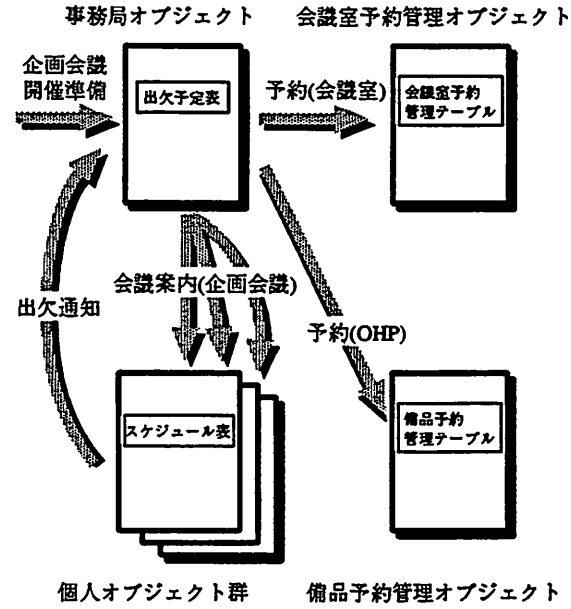

図 8: 会議開催事務処理

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。

5.4.2 受付オブジェクトの抽出

以下の 2 点の条件により、受付オブジェクトには事務局オブジェクトが当てはまる。

外部とのメッセージのやり取り 会議開催の要求メッセージを受ける。

仕事を依頼 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る。

5.4.3 受付オブジェクトが行う依頼の抽出

依頼は、以下の 3 点である。

1. 会議室予約管理オブジェクトへ会議室の予約

2. 備品管理オブジェクトへ OHP の予約

3. 会議のメンバへの会議開催案内

5.4.4 依頼からメッセージセットへの変換

受付オブジェクトが行う依頼をメッセージセットに変換する。

（1）会議室の予約

会議室予約管理オブジェクトに会議室の予約のメッセージを送れば良いので、

{ (会議室予約管理, 会議室の予約) }

となる。

（2）OHP の予約

備品管理オブジェクトに OHP の予約のメッセージを送れば良いのだが、OHP の予約ができなかった場合は、事務局オブジェクトに貸出不可のメッセージが送られるようにする場合を考える。

貸出不可のメッセージについては、"OHP の予約ができなかった場合" という条件があるので、ガードを付加すれば良い。また、事務局オブジェクト→ 備品管理オブジェクト→事務局オブジェクトとメッセージが次々と伝わって行くので、直列のメッセージセットを使えば良い。

{ (備品管理, OHP 予約),
  [OHP の予約不可] (事務局, 貸出不可) }

（3）会議開催案内

個人オブジェクトに対しては、会議案内を送り、返事として出欠通知をもらわなければならないので、

{ (個人, 会議案内), (事務局, 出欠通知) }

を送れば良い。

個人オブジェクト全員に対する会議案内メッセージの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。

{
  { (個人$_1$, 会議案内), (事務局, 出欠通知) } |
  { (個人$_2$, 会議案内), (事務局, 出欠通知) } |
  ⋮ ⋮
  { (個人$_n$, 会議案内), (事務局, 出欠通知) }
}

5.4.5 個々のメッセージの作成

メッセージセットの中に含まれていたメッセージは、会議室の予約、OHP の予約、貸出不可、会議案内および出欠通知であるので、それぞれを作成すれば良い。

5.4.6 適用結果

ここで提示したような手順を形式化した手法でモデル化がおこなえることを示した。このような方式により、例えば、"すでに予約されている会議室を譲ってもらう" という問い合わせ処理を事務局オブジェクトに追加するような拡張も容易に出来るようになる。

5.5 分散実行問題への適用

Hoop では、分散協調すべきオブジェクト群をネットワーク上のノードへ割り当てる問題や個々のオブジェクトの機能が複雑になった場合のオブジェクト分割問題を解決するために以下の 3 種類のオブジェクトを導入する (図 9)。

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。基本的に、**Hoop** ではノードオブジェクトを使い記述する。

プロセスオブジェクト 解決すべき問題が複雑すぎる場合に使用する。複数のプロセスの協調により、ノードオブジェクトが果たすべき機能を実現する。内部にサブオブジェクトを持つことができる。

サブオブジェクト 補助的なオブジェクトである。内部にサブオブジェクトを持つことができる。

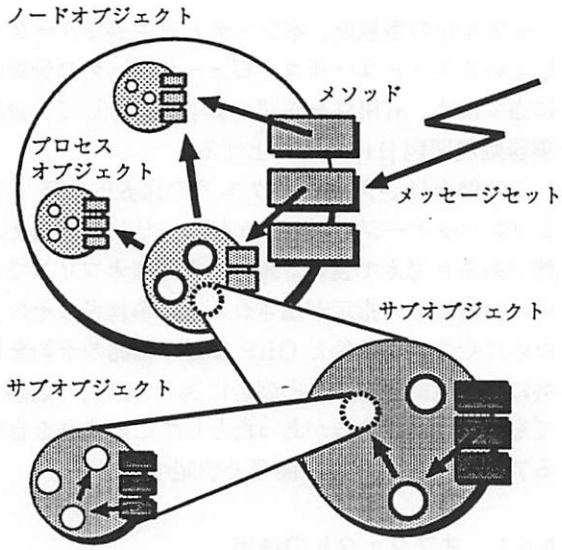

図 9: 3 種類のオブジェクト

Hoop では、オブジェクト単位に機能を分割していき問題を解決する。一番大きな分割の単位がノードオブ

ジェクトである。基本的には、ノードオブジェクトで問題を記述できる。しかし、ノードオブジェクトの機能が複雑になりすぎる場合は、ノードオブジェクト内をプロセスオブジェクトで分割することにより複雑さを解消する。ノードオブジェクトの複雑さが会議開催事務処理問題程度である場合、ふたつの実現方法が考えられる。

ひとつは、ノードオブジェクトをひとつの会議開催事務処理のようなアプリケーションとみなし、4節のモデル化の手順で示した方法で解決する。オブジェクトは、受付オブジェクトとその他のオブジェクトに分けられるが、受付オブジェクトをノードオブジェクトとし、その他のオブジェクトをノードオブジェクトの内部のプロセスオブジェクトとする。

もうひとつは、例えば会議室予約オブジェクトが複数のアプリケーションで共有される場合、それを別のノードオブジェクトにしておく方法である。この場合、実際にはふたつのノードオブジェクトが同一のノードで稼働することも可能である。

サブオブジェクトについては、ノードオブジェクトをプロセスオブジェクトで分割しても、まだ複雑である場合に使用する。

複雑なオブジェクトでも、会議開催事務処理問題程度に分割できれば、**Hoop** の分散協調型モデルで表現可能である。

5.6 プログラム例

Hoop の言語の構文の全体は当初 C++のサブセットをベースに検討していたが、現在では昨年開発された Java に注目し、そのサブセットをベースにさらにコンパクトな言語にする方針である。

以下に、会議開催問題を記述したプログラム例を示す。事務局オブジェクト (office) が個人オブジェクト (staff) に対して会議案内 (arrange()) を送り、返事として出欠通知 (replyYes() もしくは replyNo()) をもらう場合を考える。

```
class Office extends Object {
  int yes, no;
  Staff staff[ ];
  public Office()[ ]
  {
    yes = 0;
    no = 0;
    staff = new Staff[3];
    staff[0] = new Staff("abe");
    staff[1] = new Staff("baba");
    staff[2] = new Staff("chiba");
    [ self.arrange() ];
  }
  public arrange()[ ]
  {... }
  public replyYes(boolean ok)[ ]
  {... }
  public replyNo(boolean ok)[ ]
  {... }
      ⋮
}
```

6 おわりに

今回報告したモデリングツールと **Hoop** を含む **M-base** プロジェクトでは、「モデリング&シミュレーション」を支援するアプリケーション開発環境の研究開発をおこなっている。

今後は，これらの技術をモデリング&シミュレーション機能として実現するツールを開発していくが、その中で、**Hoop** を **M-base** のコンポーネントウエアおよびビジュアルプログラミングツールを支える言語として位置付けている。特に特定応用分野向きコンポーネントウェアとの連携による分散協調型システム構築技法を実現するために，言語処理系，シミュレータ，ブラウザなどを開発を行い，具体的なアプリケーションの開発に適用し，実用性を評価する.

参考文献

[1] 青山幹雄：コンポーネントウェア：部品組立て型ソフトウェア開発技術, 情報処理学会誌 Vol.37,No.1,pp.71-79 (Jan. 1996).

[2] 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.

[3] 大西 淳：ビジュアルなソフトウェア要求仕様化技法, 情報処理学会論文誌 Vol.36 No.5 P1183.

[4] 小西 裕治, 中所 武司: 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, 日本ソフトウェア科学会 第 12 回大会論文集 pp. 197-200, 1995.

[5] 柴山 悦哉, 松田 裕幸, 米澤 明憲: 並列オブジェクト指向言語 ABCL によるプログラミング, オブジェクト指向 解説と WOOC'85 からの論文, pp. 57-82 共立出版, 1985.

[6] 田村, 伊藤, 杵嶋：ドメイン分析・モデリング技術の現状と課題, 情報処理,Vol.35,No.10, pp.952-961(1994).

[7] 中所：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.

[8] 中所：エンドユーザコンピューティングのための分散オブジェクト指向設計技法, 平成６年度 EAGL 基礎・育成助成研究成果報告書, 171-182(1995).

[9] 中所：ＣＳ－ｌｉｆｅ, コンピュータソフトウエア, 11, 6, 1-2 (Nov. 1994).

[10] 中所：オブジェクト指向概念の発生学的定義に基づくソフトウェア設計技法, 情報処理学会ソフトウェア工学研究会資料,93-SE-95, 95-7, 1-8,1993.

[11] 中所: 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.

[12] 中所: エンドユーザコンピューティング－ソフトウエア危機回避のシナリオー, 情報処理, Vol.32,No.8,pp.950-960(1991).

[13] 中所: 使いやすいソフトウエアと作りやすいソフトウエアーオブジェクト指向概念とその応用ー, 電気学会雑誌,Vol.110,No.6,465-472(1990).

[14] 中所, 芳賀: オブジェクト指向型言語と論理型言語の融合方式に関する考察, オブジェクト指向, 共立出版,pp.133-146(1985).

[15] 浜：アプリケーションソフトウェア開発環境 M-base -特定分野向けアプリケーションのためのコンポーネント部品の抽出と利用-, 明治大学理工学部情報科学科、中所研究室、１９９５年度卒業論文, (Feb. 1996).

[16] 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.

[17] 吉岡：アプリケーションソフトウェア開発環境 M-base -モデリング&シミュレーションツール-, 明治大学理工学部情報科学科、中所研究室、１９９５年度卒業論文, (Feb. 1996).

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

[19] Booch,G. : Object-Oriented Design with Applications, Benjamin/Cummings (1991).

[20] Coad,P. and Yourdon,E. : Object-Oriented Design,Prentice Hall (1991).

[21] D. コールマン他 著：オブジェクト・オリエンテッド開発設計論:The Fusion Method, 横河・ヒューレット・パッカード株式会社 カストマ教育センタ訳, トッパン,1994.

[22] Eric Gamma et al., : Design Patterns, Addison-Wesley, (1995).

[23] Jacobson,I. et al. :Object-oriented Software Engineering, Addison-Wesley (1992).

[24] Jalote,P. : Functional Refinement and Nested Objects for Object-Oriented Design, IEEE Trans. Softw. Eng., Vol.SE-15, No.3, pp.264-270 (1989).

[25] Wirfs-Brock, R., Wilkerson, B. and Wiener, L. : Designing Object-Oriented Software, Prentice Hall(1990).