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た．その実現のために，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開発をマクロレベルとミクロレベルの２階層に分けて行う開発環境M-baseを研究開発している．今回，その主要ツールとして，モデリング＆シミュレーションツールとスクリプト言語の処理系をJava言語を用いて試作した．

モデリング＆シミュレーションツールは，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するものである．アイコン操作を基本としたビジュアルな機能をJava言語で実現した．モデリングは，外部インタフェース，オブジェクト間の動的なふるまい，個々のオブジェクトの内部処理の順に行うことにより，エンドユーザ主導の開発を可能にしている．

スクリプト言語では，Java言語サブセットを基本としながらメッセージの送受信方式に特徴を持たせた．すなわち，オブジェクト間の交信は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．本機能により，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った．さらに，オブジェクトをネットワーク上の任意のノードで稼働させる機能をもたせ，分散システムの構築を容易にしている．

今後は，これらのツールの機能を充実させ，両者を一体化して利用できるようにする．すなわち，ビジュアルなモデリングによってスクリプト言語プログラムの大半を自動生成できるようにする．

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に記載.

■今後の課題

・第１ページの「概要」に記載.

■有形成果物（論文を除く）

・モデリングツールとスクリプト言語処理系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

■実用化シナリオ

・まず自分たちで利用するアプリケーションの開発に適用し，実用化を推進する.

■学会発表リスト

・T. CHUSHO, Y. KONISHI and M. YOSHIOKA : M-base : An Application Development Environment for End-users Computing based on Message Flow, APSEC'96 (Asia-Pacific Software Engineering Conference), pp.366-375 (Dec. 1996).

・中所武司，小西裕治，浜巨人，吉岡大生：「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M-baseの開発技法、情報処理学会ソフトウェア工学研究会資料、96-SE-109, 109-2, pp.9-16（May.1996）.

・小西裕治，中所武司：分散協調型アプリケーションのためのオブジェクト指向分析・設計言語Hoopの設計とその記述実験，情報処理学会オブジェクト指向 '96 シンポジウム，87-94 (July 1996).

・松本光由，中所武司：「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M-baseにおけるモデリング＆シミュレーション技法，情報処理学会ウィンターワークショップ・イン・松山，（Jan. 1997）.

・T. Chusho : wwHww : An Application Framework of Multi-organizational Office Network Systems, Memoirs of the Institute of Science and Technology, Meiji University, Vol. 35, No.1, (1996).

・藤原克哉，小林芳幸，中所武司：分散環境における簡易対話ツールV-Saloon（仮想サロン）の開発と適用評価，情報処理学会第53回大会，4/21-22 (Sep. 1966).

・斉藤裕樹、松本光由、安斎恵、中所武司 ：多組織間オフィスネットワークシステムにおける分散アプリケーションフレームワークwwHww，利用者指向の情報システムシンポジウム，情報処理学会，29-36 (Dec. 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．このような新しい動向に対応して，コンポーネントウェアやビジュアルプログラミングに代表されるような新しい開発技法が普及しはじめている．

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，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．

”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**[6] を開発中である．

今回，マクロレベルで利用するツールとして，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&シミュレーションツールを試作した．またミクロレベルで利用するツールとして，スクリプト言語の基本機能を実現する処理系を試作した．いずれも Java 言語で実装した．

本報告では，２章では本研究の目的と対象，３章ではモデリングプロセスの概要，４章では開発環境，５章ではモデリング&シミュレーション，６章では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，エンドユーザの範囲は広いが，本論文では，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基幹業務担当者および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業務の専門家を対象とし，特に今後急速に増大すると思われる後者の方により重点を置く．

2.2 対象ソフトウエア

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」という観点から，オフィスでの業務用アプリケーションを中心にグループウェアやワークフローシステムなども対象とする．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，問題領域を分析してドメインモデル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，そのモデル上でシミュレーションして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．

この時，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ことも重要である．

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．

3 モデリングプロセスの概要

3.1 2階層モデル

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の分野では，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，次に示すようなマクロモデルとミクロモデルの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1に示す．詳細は文献[3]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]
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]があるメッセージを受信した後，メッセージの列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．

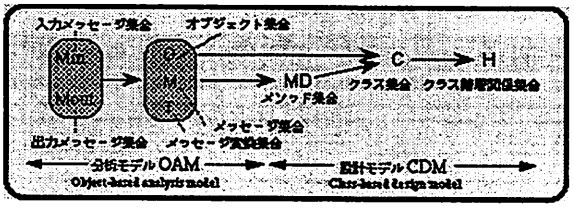

図 1: M-baseのモデリングプロセス

4 開発環境の概要

M-baseの開発環境とアプリケーション・アーキテクチャの関係を図2に示す．開発環境は主に次のような構成要素からなる．

- モデリング&シミュレーションツール
- スクリプト言語
- UIビルダー
- コンポーネントビルダー
- 共通プラットフォーム

これらのツールを用いて開発されるアプリケーションは大まかには次の3つの部分から構成される．

- ユーザインタフェース
- モデル
- コンポーネントウェア

モデルはアプリケーションに固有の処理を行う本体である．動的モデル（ドメインモデル）に対応するOAMの部分をモデリング&シミュレーションツールを用いて作成する．静的モデル（オブジェクトモデル）に対応するCDMの部分をスクリプト言語を用いて作成する．

ユーザインタフェースはそのアプリケーションのユーザとの対話処理部分であり，モデルと独立させることにより，クライアント／サーバシステムなどのシステム構成，あるいはクライアント端末のマルチプラットフォーム化が容易になる．その構築ツールとしてUIビルダーがある．

コンポーネントウェアには分野に共通の基本部品と特定業務分野向きの部品がある．後者が充実してくるとエンドユーザによるアプリケーション構築が容易になる．共通プラットフォームはミドルウェアと基本ソフトウェアの部分で，オープンシステムや部品の流通の観点からは特に分散オブジェクト管理機能などが重要である．

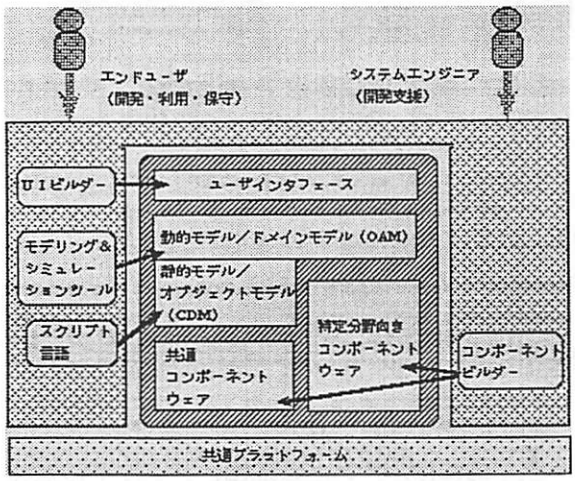

図 2: M-base の開発環境（外側）とアプリケーション・アーキテクチャ（内側）

5 モデリング&シミュレーション

5.1 1オブジェクト1業務の原則

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．

5.2 基本機能

モデリング&シミュレーションツール [5] 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，オブジェクトとオブジェクト間のメッセージパッシングで構成される．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．

M-base におけるモデリングは次の手順で行なわれる．

1. 外部インタフェースのモデリング
   (a) 外部入力の抽出
   (b) シナリオの記述
   (c) 外部インタフェースの定義
2. 動的な振舞いのモデリング
   (a) オブジェクトの抽出
   (b) メッセージの抽出
3. オブジェクト内部のモデリング
   (a) メンバ（属性）の決定
   (b) メソッドの意味定義の記述
   (c) メソッド内部のスクリプティング

5.3 外部インタフェースのモデリング

まずはじめにシステムの利用方法を明確にするために，OOSE[11] と同様の一般的な手法をとる．すなわち，システムの利用者（アクタ）と利用方法（ユースケース名）から外部入

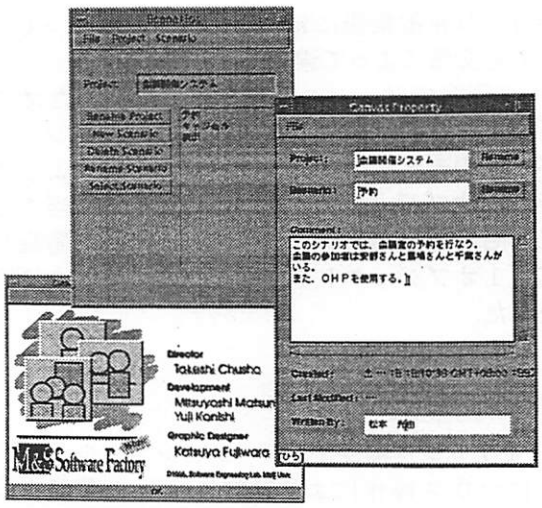

図 3: 会議開催シナリオの記述例

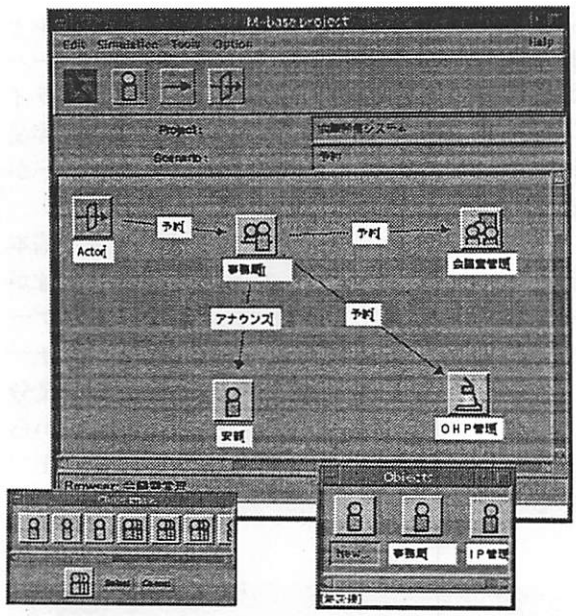

図 4: 会議開催の業務フローのモデリング

力を抽出する．次に各々の外部入力から開始される機能概要（シナリオ）を記述する．そして最後に外部インタフェース，即ち図1における Min， Mout を決める．

例えば， M-base を用いて会議開催システムを開発する場合を考えてみると，会議開催者，会議出席者，等のアクタが存在し，会議開催者に対しては，会議の準備やキャンセル等のシナリオが存在する．図3は，会議開催シナリオの詳細を表示しているウィンドウである．

5.4 動的な振舞いのモデリング

このフェーズでは，各シナリオに対するシステムの動的な振舞い，即ち，インスタンスベースのメッセージのやりとりをモデリングしていくことにより，オブジェクトとメッセージを抽出する．オブジェクトの抽出は，「１オブジェクト１業務」の原則に従って行なわれる． メッセージは，業務間で通信し合うための手段（書類，電話，口頭連絡等）に対応させて抽出する．

この部分で用いるツールの機能は，ドローツール風の視覚的にわかりやすいインタフェースとしている．図4がそのウィンドウであり，左上のメニューの下の４つのボタンがそれぞれ左から，選択，オブジェクト，メッセージ，アクタのモード指定ボタンである．

それぞれのモードを選択し，アクタからのメッセージ (Min) の記述，オブジェクトのアイコンの配置などが行なえる．また，メッセージは，送信側と受信側を指定し作成する． その他，基本的な編集機能も備えている．

5.5 オブジェクト内部のモデリング

基本的には，メンバの抽出，メソッドの意味定義，メソッド内部の実装の３つのステップがある．このフェーズには，コンポーネントの組み立てや，分散アプリケーションの実現などの多岐にわたるが，現段階では，単一のオブジェクトの詳細を決めていくウィンドウを用い，これら３つの作業を行なっていく．

図5は，会議開催システムの中の事務局オブジェクトの内部を記述しているウィンドウの例である． このウィンドウには，次のような情報が表示され，定義できる．

- オブジェクトのアイコン
- オブジェクトの名前
- メンバのリスト
- メソッドのリスト
- 各メソッドの意味定義

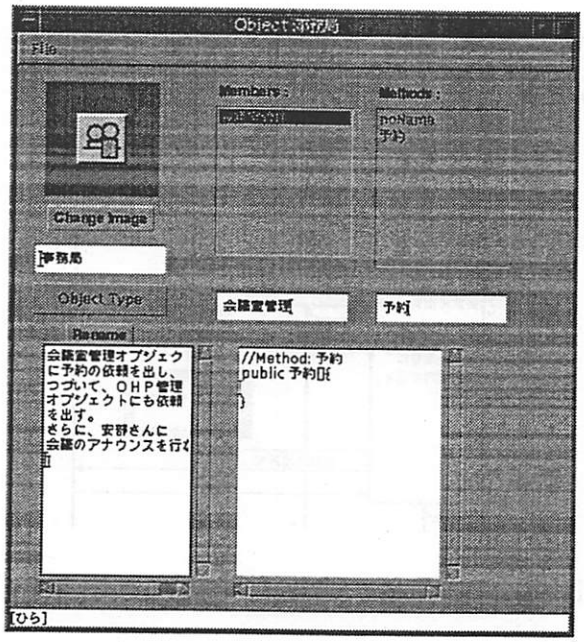

図 5: 事務局オブジェクトの内部の記述例

- 各メソッドのスクリプト

この中で特にメソッドの意味定義は，モデリングからシミュレーション，およびこれらのマクロモデルからミクロモデルへの橋渡しとして重要である．

すなわち，ドメインモデルの作成段階において，メソッド名だけではシミュレーション時に動作の確認が難しいと思われる．また，メソッド名から内部のスクリプティングに移っていくのには大きなギャップがある．そこで，メソッドの意味定義をモデリング時に記述することにより，この問題を解決している．

以下では，会議開催システムの例を用いて，メソッドの意味定義について説明し，シミュレーション時の利用法と有効性について述べる．

会議開催システムの主な機能は，

「会議開催要求をうけると会議室の予約と OHP などの備品の予約を行なうと共に，会議開催通知を出席予定者に送り，出欠の返事を返してもらう」

というものである． この会議開催システムのドメインモデルを図 6に挙げる．このモデルにおける，事務局オブジェクトの “会議開催準備”

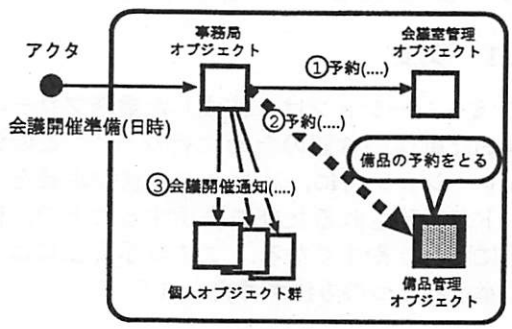

図 6: ドメインモデルを用いたシミュレーション

メソッドの意味定義は，次のようになる．

事務局オブジェクト：会議開催準備 (日時)

1. 会議室の予約要求を出す．
   - 会議室の予約がとれなかったら，その旨を会議開催者に通知する．
2. 備品の予約要求を出す．
   - 備品の予約がとれなかった場合は，会議主催者にその旨を通知し，会議の取消を行なうか確認する．
3. 会議案内を出席予定者に送る．

メソッドの意味定義は，次の２つの記述からなる．

基本系列 上の例で， 1. から 3. の各１行目がこれにあたる． 基本系列は，メソッドの処理の概要を示すものであり，最初にこれを記述する．

例外処理 上の例では，●印部分がこれにあたり，業務フローにおける例外処理を示す．例外処理は，早い段階で発見でき，気づいた時点で記述しておく．

5.6 シミュレーション機能

5.6.1 概要

シミュレーションは，作成した業務フローの基本的な動作の確認のために行なう． このシミュレーション時に，メソッドの意味定義をメソッドが起動されるたびに表示することで，視覚的にわかりやすくなる．この表示方法には，次に挙げる2つの方法が考えられる．

1. ドメインモデルをそのまま用いる方法
2. 事象トレース図を用いる方法

前者の方法は，ふきだし表示などにより，作成したモデル上で動作の確認が行なえるため，直観的に理解しやすい． 図6がその例であり，備品管理オブジェクトの予約メソッドを実行中であることを示している．

後者の方法は，システムが動作を開始すると，起動されたメソッドの定義内容が事象トレース図上に表示されていくので，オブジェクト間通信の全体のシーケンスを見渡すことが可能である．図7がその例であり，事務局オブジェクトの会議開催準備の処理の中で，会議室管理オブジェクトと，備品管理オブジェクトにメッセージを送り，現在，備品の予約処理を行なっているところである． 実行済みの業務フローは，太い線で表示されている．

これらの表示方法を併用して使うことにより，メソッドの意味定義の利点を充分発揮できると考えている．

5.6.2 アニメーション

図8右下の “Start” ボタンを押すと，シミュレーションを開始する． さらに， “Next” ボタンを押すと，業務フローが逐次流れていく．図8では，呼出中のメッセージの色が変わり太い線で表示されている．現在，事務局オブジェクトが，備品管理オブジェクトにメッセージを送信しているところである． また，メソッドの意味定義の表示については，事象トレース図を用いた方法を実装をした．ドメインモデルをそのまま用いる方法については，まだ実装途中である．図8の左側の上が事象トレース図を用いた例であり，メッセージが送信されるごとに，メソッドの意味定義表示用のエリアが次々生成され，時系列に従って配置されていく．そのエリアの中に，オブジェクト内部のモデリング時に

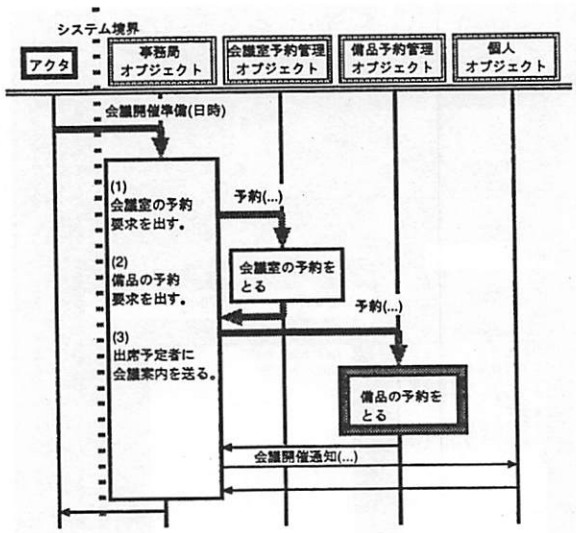

図 7: 事象トレース図を用いたシミュレーション

記述したメソッドの意味定義が埋め込まれて表示されている．

6 スクリプト言語 Hoop

6.1 設計思想

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，オブジェクト指向言語 **Hoop**[2] を開発中である．

本言語は，分析・設計段階で使用することを目的としており，以下のような特徴を有する．

- 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．

6.2 分散協調型モデルの表現方法

6.2.1 メッセージセット

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

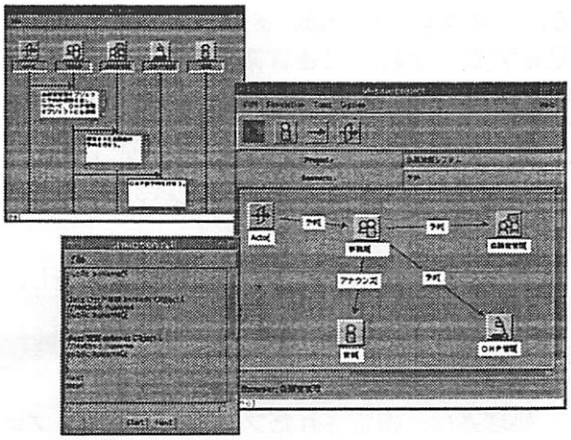

図 8: シミュレーションの画面例

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メタな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．

6.2.2 メッセージセットの構文

メッセージセットの構文を以下に示すが，言語の詳細は文献 [2] に詳しい．

$$
\begin{aligned}
M &::= M' \| [cond] M' \\
M' &::= M_s \| M_p \| X \\
M_s &::= \{M, M, \ldots, M\} \\
M_p &::= \{M | M | \ldots | M\} \\
X &::= (obj, msg)
\end{aligned}
$$

obj はオブジェクト， msg はメッセージ， $cond$ は条件， $\alpha \| \beta$ は α， β のどちらかを意味する．

6.2.3 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．

例えば，以下のような直列にメッセージを送る場合を考える．

例) 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．

このような直列のメッセージセットは以下のように記述する．

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

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．

6.2.4 並列のメッセージセット

一方，並列にメッセージを送る場合は以下のように記述する．

例) 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．

このような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．

6.3 分散システムの実行方式

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．

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な場合に使用する．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．ただし，ノードオブジェクトやプロセスオブジェクトと異なり，逐次処理に限られる．

オブジェクト単位に機能を分割していくときの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な場合は，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．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．

6.4 言語処理系（試作版）

6.4.1 機能概要

今回開発した言語処理系は次のような機能を支援する．

- マクロレベルの記述
- ミクロレベルの記述
- シミュレーション

マクロレベルでは，メッセージセットを用いてメッセージのフロー (直列，並列，ガード) を記述することにより，クラスの外部仕様が決まる．ミクロレベルでは，メソッドの内部仕様を記述する．なお，記述言語としては，M-base のオブジェクト指向モデルとの親和性を重視して，基本言語として Java 言語を採用し，そのサブセットに Hoop 固有の機能を付加する方式とした．そしてこれらの仕様をインタプリタで実行 (シミュレーション) する．

プログラムは，ひとつのファイルに記述し，次のようなコマンドで実行を指示する．
java hoop.Main “ファイル名”
“クラス名” “メソッド名” “引数”

処理系は，指定されたファイルのソースプログラムを解析し，木構造に変換し，必要なインスタンスを生成し，指定されたクラスの指定されたメソッドから実行を始める．

分散システムの構築機能としては，3 種類のオブジェクトのうち，ノードオブジェクトとサブオブジェクトを実現した．サブオブジェクトを生成するには，プログラム中で次のように宣言する．

new クラス名 ()

ノードオブジェクトを生成するには，プログラムの起動時に稼働させたいノード側で次のように指定する．
java hoop.Main -n “ノード名” “ファイル名”
“クラス名” “メソッド名” “引数”
ノード名は，*//hostname/name* とする． *name* については任意の名前でよい．

ノードオブジェクトの参照が必要な時は，プログラム中で次のように指定する．

bindクラス名()[“ノード名”]

メッセージセットは今回の実装では以下のように記述する．6.2.2 で定義した構文規則を少し変更した．

基本単位 (obj, msg) → obj.msg(引数)
直列のメッセージセット → [M, M, ..., M]
並列のメッセージセット → [M | M | ... | M]
ガード → (*expr*)

M は， obj.msg(引数) または直列あるいは並列のメッセージセットである．

6.4.2 記述例

会議開催システムの記述例を示す．マクロモデルでは，オブジェクトを生成し，メッセージセットによるフローの記述をする．必要なオブジェクトは，事務局オブジェクト，会議室予約管理オブジェクト，備品管理オブジェクト，個人オブジェクト (3 人分用) である．

以下はオブジェクト生成の記述部分である．

```
class Main {
    public main(String args[]){
        Office office;
        Reservation room_mgr;
        Reservation ohp_mgr;
        Staff abe;
        Staff baba;
        Staff chiba;

        office = new Office();
        room_mgr = new Reservation();
        ohp_mgr = new Reservation();
        abe = new Staff();
        baba = new Staff();
        chiba = new Staff();
            :
    }
}
```

事務局オブジェクトが会議開催準備の arrange メッセージを受けとった時に実行する業務は以下のようなメッセージフローで記述する．

```
class Office {
    public arrange(){
        [ room.reserve(), this.reply(boolean) ];
        [ ohp.reserve() ];
        [ staff_a.reserve() | staff_b.reserve() |
            staff_c.reserve()];
    }
}
```

このようなマクロレベルの記述は，最終的にはモデリング&シミュレーションツールを用いて作成したマクロモデルから自動生成されるものであるが，現段階では連携していない．

ミクロモデルでは，メソッドの内部仕様を記述する．会議室予約管理オブジェクトの予約メソッド reserve() は，以下のようになる．

```
class Reservation {
    public reserve(Date d)[ next(boolean) ]{
        int i;
        boolean result;

        result = true;
        for (i = 0; i < size; i = i + 1){
                if (d.equal(schedule[i]))
                result = false;
        }
        if (result){
            schedule[size] = d;
            size = size + 1;
            next(true);
        }
        else
            next(false);
    }
}
```

next(...) を実行すると，残りのメッセージセットが呼ばれる．

```
[ room.reserve(), this.reply(boolean) ];
```

の場合， room は reseve() を受け取り， next(...) を実行した時点で

```
[ this.reply(...) ];
```

が呼ばれる．

6.4.3 実装

Hoop は，Java で記述されたインタプリタ言語である．最終的には，インタプリタはモデリング&シミュレーションツールと統合して利用される．そしてシステムの完成後はコンパイラで最適なコードを生成するのが望ましい．

プログラムは全体で 6000 ステップ，クラス数は 55 クラスである．主要な処理のプログラムサイズを以下に示す．

内容内容	クラス数	ステップ数
字句解析	1	400
構文解析	1	1600
構文木	16	1200
クラス	1	100
メッセージセット	9	900
メソッド	2	200
クラスライブラリ	13	300

構文木とメッセージセットのクラスなどは重複しているので，正確な数字ではない．

ネットワーク部分の実装は，ORB の技術を使用している．Java で使えるものとして RMI, HORB[10] が考えられたが，記述の簡単な RMI を使用した．

7 おわりに

「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 M-base の開発技法について述べた．特にモデリング＆シミュレーションツールとスクリプト言語処理系については試作を終了している．今後は，これらのツールを改良し，統合すると共に，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, 37, 1, 71-79, 1996.

[2] 小西 裕治, 中所 武司: 分散協調型アプリケーションのためのオブジェクト指向分析・設計言語 Hoop の設計とその記述実験，情報処理学会オブジェクト指向シンポジウム oo'96, 朝倉書店, 87-94, 1996.

[3] 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, 104-4, 25-32, 1995.

[4] 中所 武司，小西 裕治，浜 巨人，吉岡 大生：「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 M-base の開発技法，情報処理学会ソフトウェア工学研究会資料，96-SE-109, 109-2, 9-16, 1996.

[5] 松本光由，中所武司：「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 M-base におけるモデリング＆シミュレーション技法，情報処理学会：ワークショップ・イン・松山, 27-28, 1997.

[6] T. Chusho, Y. Konishi and M. Yoshioka : M-base : An Application Development Environment for End-users Computing based on Message Flow, APSEC'96 (Asia-Pacific Software Engineering Conference), 366-375, 1996.

[7] Coleman, D. et. al. : Object-Oriented Development, The Fusion method, Prentice Hall, 1994.

[8] E. Gamma, R. Helm, R. Johnson and J. Vlissides : Design Patterns, Addison-Wesley, 1995.

[9] Gosling, J. and McGilton, H. （日本サン 訳）：Java 言語環境 技術白書，日本サン・マイクロシステムズ，1995.

[10] Satoshi Hirano: HORB Flyers Guide, http://ring.etl.go.jp/openlab/horb/ 1996.

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

[12] D. E. Monarchi and G. I. Puhr : A research typology for object-oriented analysis and design, Comm. ACM, 35, 9, 35-47, 1992.

[13] Rubin, K. and Goldberg, A. : Object Behavior Analysis, Comm. ACM, 35, 9, 48-62, 1992.

[14] Rumbaugh,J.et al.: Object-Oriented Modeling and Design,Prentice Hall, 1991.

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