メッセージフローに基づく分散協調型アプリケーション開発技法

中所武司[†] 小西裕治[††] 松本光由[†††]

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．そのためには，業務の専門家が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が必要であるという観点から，ドメインモデルにおける1業務をオブジェクト指向概念に基づく計算モデルの1オブジェクトに対応させるような開発技法を実現した．従来のオブジェクト指向分析設計技法のように最初にクラス間関連図を作成する方法は業務の専門家には難しいので，インスタンスベースのモデル構築を基本とした．エンドユーザはまずビジュアルツールを用いてメッセージフローに基づく業務モデルの構築を行い，シミュレーション実行により動作確認を行う．そこで導入した業務コンポーネントのうち，既存のもの以外はスクリプト言語で構築する．そのためにビジュアルな業務モデルから自動変換される実行可能プログラムの記述言語をコンポーネント記述言語と同じにして，動的モデル上でのシミュレーション実行を容易にした．このスクリプト言語には，オブジェクト間の交信をメッセージセット単位で行う方式を導入して，柔軟なワークフローを実現した．

Distributed-application Development Based on Message Flows

TAKESHI CHUSHO,[†] YUJI KONISHI[††] and MITSUYOSHI MATSUMOTO[†††]

Explosive increase in end-user computing on distributed systems requires that end-users develop application software by themselves. One solution is given as a method that one task in a domain model of cooperative work corresponds to one object in a computation model based on an object-oriented model. An end-user builds a domain model by using a visual tool at the first step, and then confirms the system behavior by simulation of the model. If some business components among the domain model have not implemented yet, these components should be developed. A script language is used for describing a new component. In addition, this language is used as a dscription language for an executable program which is automatically generated from a domain model while focusing on message flow. Therefore this language has the specific feature of describing a message flow as a message set for implementation of flexible work flow.

1. はじめに

近年，インターネットやイントラネットに接続されたパソコンの普及とともに，オフィス業務の効率化という観点から，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．たとえば，基幹業務システムのように投資に対する効果が明確なものは，従来のように情報処理の専門家が開発すればよいが，小さな部門あるいは個人の業務を対象とするものや頻繁に機能変更が発生するものには従来の開発方法はなじまない．このようなアプリケーションは，業務の専門家が自ら開発し，自ら保守を行うのが望ましい．

このような新しい動向に対応して，90年代半ばからコンポーネントウェア[1)~3)]やビジュアルプログラミング[4)]に代表されるような新しい開発ツールが実用化されはじめている．この分野では，クラスからのインスタンス生成という概念を意識させないで，すぐに動作するインスタンスオブジェクトをコンポーネントとして扱うものも多い．コンポーネントウェアの中でも特定分野の業務に固有のものはビジネスオブジェクト[5)]と呼ばれている．しかしながら，現時点では，これらのツールが対象とする業務は，共通性が高い典型的な処理に限られている．多種多様な業務をモデル化の段階から支援してアプリケーションを構築するようには

† 明治大学理工学部情報科学科
Department of Computer Science, Meiji University
†† 日立ソフトウェアエンジニアリング株式会社
Hitachi Software Engineering Co., Ltd.
††† 株式会社日立製作所
Hitachi, Ltd.

なっていない．

一方，モデリングを重視した開発技法として，90年前後からオブジェクト指向分析・設計技法[6),7)]が注目されてきた．しかしながら，これらの開発方法論は情報処理の専門家が使用することを前提にしており，その適用には専門的なスキルが要求される．分析，設計の初期段階でクラスの抽出やクラス間の関係の定義が必要なもの[8)~12)]が多く，エンドユーザ主導の開発には向かない．

我々は，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開発方法論が必要であるという認識から，「ドメインモデル ≡ 計算モデル」という基本コンセプト[13)]を設定した．これは，ドメインモデルを計算モデルとして扱うことにより設計やプログラミング作業を回避しようとするものである．

このコンセプトを実現するために，プロトタイピングアプローチを支援する開発環境 M-base[14)]を開発した．主な機能としては，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な振舞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[15)]と，業務モデルの自然な表現としてのメッセージセット機能を有するスクリプト言語[16)]がある．

本論文では，2章で開発方法論と環境，3章でモデリング&シミュレーション，4章でスクリプト言語，5章で実行方式について述べ，6章で OMG や WfMC における標準化活動に言及しながら結果について考察する．

2. 開発方法論と環境

2.1 研究の目的と対象

一般に，エンドユーザの範囲は広いが，本研究では，たとえばユーザ企業のエンドユーザ部門に所属し，市販のアプリケーションパッケージを利用しているような人たちを対象とする．

対象ソフトウェアとしては，オフィスでの業務を中心にワークフローシステムやグループウェア[17)]などの分散協調型システムを含む．ただし，市販のパッケージでは対応できないような，きめ細かい機能の実現が要求される場合を想定する．規模的には小規模のアプリケーションを想定することになるが，ネットワーク接続することによりシステムとしては大規模化することもある．

このようなソフトウェアは，最初に要求仕様を明確にすることが難しく，稼働後の頻繁な機能変更も必要であるため，従来の基幹業務のように情報システム部門や外注先に開発・保守を委託することは適さない．エンドユーザ主導の開発・保守形態が必須である．

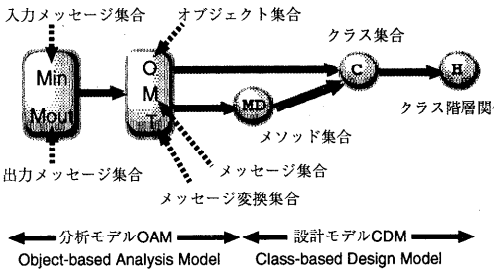


図1 M-base のモデリングプロセス
Fig. 1 Modeling process of M-base.

そこで，本研究では，問題領域を分析してドメインモデルを構築し，その妥当性をシミュレーション実行により検証した後，そのまま実用システムとして利用するという方法をとる．このとき，特定分野向きのコンポーネントウェアを利用すれば問題領域のモデリングが容易になる．

2.2 モデリングの基本プロセス

従来のオブジェクト指向分析・設計技法では，最初にクラスの抽出やクラス間の関係などを定義するものが多かった．これは，従来の情報システムの分野では，外部環境の変化に対してシステムの機能が敏感に影響を受けるのに対して，データ構造の方が安定しているという経験則[12)]に対応している．

本研究では，先に述べたような研究対象の特徴から，クラス間の関連に最初に注目する従来の手順とは逆に，まずインスタンスベースの動的構造を明確にした後，必要に応じて静的構造を定義するという手順をとることにした．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振舞いを記述するために，図1に示すような2段階のモデル化を実現した．すなわち，第1段階ではシステム全体の動的な振舞いをオブジェクト間の協調関係で規定する動的モデルを作成する．次に，第2段階では個々のオブジェクトの振舞いをクラス定義で規定する静的モデルを作成する．

最初に構築されるインスタンスベースの動的モデルはオブジェクトの集合 O={o[i]}，メッセージの集合 M={m[i,j,n]}，メッセージ変換の集合 T={t[r]} の3つ組 {O, M, T} で規定される．メッセージ変換とは，動的モデルに含まれる 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を m[i,j,n] とすると，o[j] が m[i,j,n] を

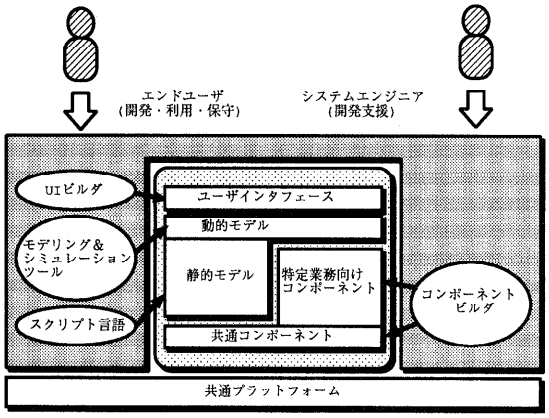


図2 M-baseの開発環境（外側）とアプリケーション・アーキテクチャ（内側）

Fig. 2 Development environment of M-base (the outer part) and application architecture (the inner part).

受信後に別のメッセージ列を送信するという関係を

$$t[r] : m[i,j,n] \rightarrow m[j,k_1,n_1],\ m[j,k_2,n_2],\ \ldots$$

と表現したものである．

次に，この動的モデルから静的モデルを作成する．静的モデルはクラスに基づいて構築されるので，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 {MD, C, H} で規定される．

このモデリングプロセスの形式化は文献14)に詳しい．なお，後で詳細に述べるが，エンドユーザはこのような形式的なプロセスを意識する必要はない．

2.3 モデリングとアーキテクチャ

本研究ではエンドユーザ主導の開発を目的としているので，実際のモデリングは業務コンポーネントの組合せを基本としている．計算モデルの観点では，開発対象のアプリケーションのソフトウェア・アーキテクチャ[18]が重要である．ここではドメインモデルを計算モデルと見なすアプローチをとっているので，以下のような3つの部分から構成されるアーキテクチャを想定した．

- ユーザインタフェース
- モデル
- コンポーネントウェア

これは，M-baseの開発環境を示す**図2**の内側の部分に対応する．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り，動的モデルと静的モデルからなる．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る．モデルと独立させることにより，クライアント/サーバシステムの構築，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．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．

これらの3つの構成要素をプレゼンテーション層，ファンクション層，データ層に対応させることにより，3層のクライアント/サーバのアーキテクチャと見ることもできる．ただ，コンポーネントには，種々の業務対応のものを想定しているので，概念としては広くなっている．このコンポーネントをネットワーク上の任意のノードに割り付けて稼働できるようにすることで，分散協調型のアプリケーションを実現している．

2.4 開発環境

M-baseの開発環境は，図2の外側部分に示すような4つのツールからなる．

モデリング&シミュレーションツールは動的モデルの作成に用いる．動的モデルはオブジェクト指向の分散協調型モデルとして表現するので，オブジェクト間のメッセージフローの定義が重要な機能といえる．

スクリプト言語は，静的モデルの定義に用いるものである．利用方法としては，動的モデルから自動生成される場合とコンポーネントの処理を明示的に記述する場合がある．

UIビルダは，部品をドラッグ&ドロップで配置し，属性に値をセットしてユーザインタフェースを構築するもので，基本機能は電子フォームなどの作成ツールと同じである．本システムに固有の機能としては，モデリング&シミュレーションツールとの連係処理[19]がある．モデリング時に入手したユーザインタフェース情報をUIビルダに渡して，ユーザインタフェースの構築をナビゲーションする．

コンポーネントビルダはコンポーネント作成用のエディタである．現時点では，スクリプト言語を用いてコンポーネントを記述するため，エンドユーザにもそれなりのスキルが要求される．

3. モデリング&シミュレーション

3.1 モデリングの手順

3.1.1 概　　要

エンドユーザによるアプリケーション開発の基本は，オブジェクトとメッセージフローで構成される動的モデルの作成である．モデリング&シミュレーションツールを用いたマウス操作により，**図3**に示すような動的モデルを構築する．

既存のビジュアルプログラミングツールでも，オブジェクト間をリンクで結んでアプリケーションを構築していくものがある．しかし，それらのツールはデー

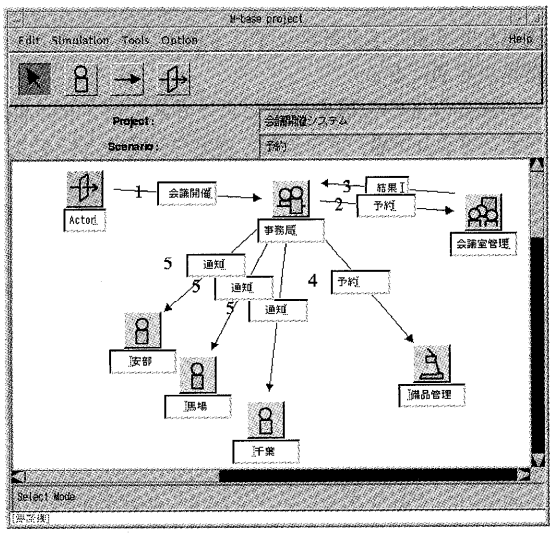


図3 動的モデル
Fig. 3 An example of dynamic model.

タベースから読み込んだデータをグラフ表示するといったような典型的な処理には向いているが，多種多様な業務をモデル化の段階から支援してアプリケーションを構築するようにはなっていない．本研究では，モデル化の各段階でシミュレーションによる動作確認をしながら次第に詳細化できるようにした．

例題として会議開催システムを取り上げ，実際の開発手順に従ってモデリング&シミュレーション技法を説明する．事務局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プロジェクタなどの備品の予約を行い，会議開催通知を出席者に送る，というオフィス業務の自動化を想定する．

3.1.2 外部仕様の作成

はじめにシステムの利用方法を明確にするために，システムの利用者と機能概要を記述する．すなわち，システムの利用者（アクタ）を抽出し，各々の外部入力から開始される機能概要（ユースケース[20]）を記述する．そして外部インタフェース，すなわち図1における Min，Mout を決める．ここでは，会議開催者が利用する会議開催に関する機能概要を記述する．外部インタフェースは「日時を指定して会議の準備を依頼する」となる．

3.1.3 動的モデルの作成

次にシステムの動的な振舞いをインスタンスベースのメッセージのやりとりとしてモデリングしていく．オブジェクトやメッセージの抽出は，以下の方法で行う．

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

(2) 1人また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見なす．

(3) 日常的業務における協調作業はメッセージの送受信として実現する．

1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，メタファベースのモデル化を行う．実世界と同じように1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」の割当てを原則とした．

業務のモデリングを行うエンドユーザの観点では，実世界で1人で複数の業務を担当している場合にはそのままモデル化する方が簡単に見えるかもしれないが，実際にはワークフローに対応するメッセージフローは複雑になる．1オブジェクト1業務の割当て原則は，実世界において十分な能力を有する業務の専門家が多数存在するという状況下でワークフローが最も簡潔になるように業務を割り当てる場合に相当し，エンドユーザには分かりやすいものである．

90年代前半に提案されたオブジェクト指向分析技法のように，業務仕様に現れる名詞をオブジェクト候補としてあとで不適切なものを除くような方法[9),10)]ではオブジェクト候補が多くなる．適切なオブジェクトの選択にスキルが要求されることになるため，エンドユーザ主導開発には適さない．

またコンポーネントの再利用という観点では，本論文の1オブジェクト1業務の原則はビジネスオブジェクト[5)]の抽出を容易にするものと思われる．

会議開催システムのモデルの例を図3に示す．事務局，会議室管理，備品管理，個人の4種類のオブジェクトをアイコンで表示し，お互いにやりとりするメッセージをアイコン間の結線で表示している．メッセージに付けられている番号は直列や並列のメッセージ送信順を示し，後述するスクリプト言語のメッセージセットの自動生成に利用する．

3.1.4 オブジェクト内部の詳細化

次に必要に応じて，個々のオブジェクトに固有のメンバ（属性）の定義，メソッドの意味定義，メソッド内部のスクリプティングを行う．

特にメソッドの意味定義は，動的モデルから静的モデルへの橋渡しとして重要である．動的モデルの作成段階で，メッセージフローによるメソッドの呼び出し関係では表現できない処理概要をメソッドの意味として定義しておくことにより，スクリプト言語での詳細

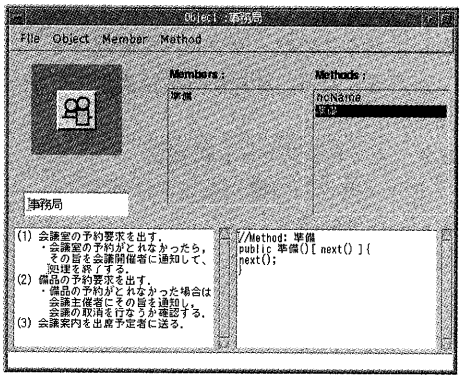


図4 事務局オブジェクトの内部の記述例
Fig. 4 An example of object definition.

化が容易になる.

たとえば,図3の事務局オブジェクトの会議開催メソッドにおいて,会議室予約ができなかった場合は会議開催者に通知して終了するという例外処理が決まっていれば,それをメソッドの意味定義として記述しておく.図4は,図3の事務局オブジェクトをダブルクリックして開いたウィンドウである.左下のボックスで準備メソッドの意味定義を入力している.右下のボックスには,準備メソッドのスケルトンコードが表示されている.

3.1.5 シミュレーションによる検証

シミュレーションは,作成した業務フローの動作確認のために行う.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,視覚的に分かりやすくなる.

この表示方法には,次にあげる2つの方法が考えられる.

(1) 動的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[14]などにより,ユーザが作成したモデル上で動作の確認が行えるため,直観的に理解しやすい.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意味定義の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流れを確認しやすい.たとえば,図5の例は,startボタンをクリックすることにより外部から事務局オブジェクトにメッセージが送られ,さらにnextボタンをクリックすることにより事務局オブジェクトから会議室管理オブジェクトにメッセージが送られ,結果が返されたところである.

3.2 コンポーネントの開発

3.2.1 コンポーネントの粒度

エンドユーザは情報処理技術に関する専門的な知識

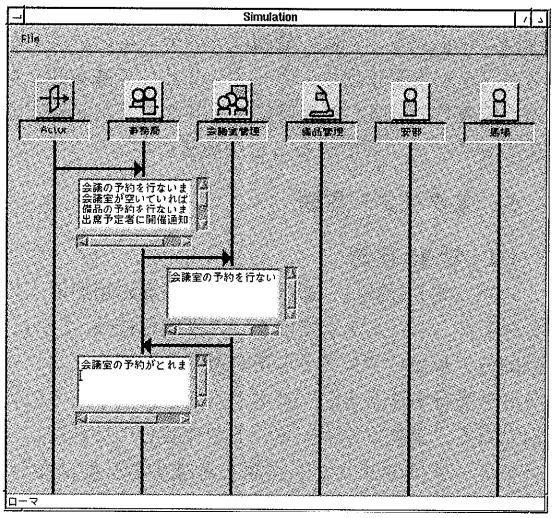


図5 事象トレース図を用いたシミュレーション
Fig. 5 An example of simulation on an event trace diagram.

が十分ではない場合が多いので,エンドユーザ主導でのコンポーネントの抽出,作成,再利用においては,その粒度が重要である.すなわち,一般に大きい粒度の業務コンポーネントは扱いが容易である反面,再利用における柔軟性が乏しくなる.一方,業務間に共通の小さい粒度のコンポーネントは適用範囲が広くなる反面,抽象データ型レベルのクラスライブラリのようにプログラミングの知識が要求されるので利用技術が難しくなる.

そこで,最近では従来の部品の概念を発展させた技術[21),22)]として,コンポーネントウェア,アプリケーション・フレームワーク[23)],デザインパターン[24)]などが注目されており,見かけ上の部品の粒度を大きくして使いやすくする役割を有する.

M-baseでは,エンドユーザ主導の開発という観点から,業務モデルのオブジェクトに対応した適切な業務コンポーネントを選択することが重要である.

3.2.2 コンポーネントの抽出と作成

業務コンポーネントの抽出は,すでに述べたように1つの業務を擬人化したオブジェクトに割り当てるというメタファベースのモデル化の中で行う.

そのコンポーネントの作成にはスクリプト言語を用いるため,エンドユーザにもそれなりのスキルが要求される.そこで複雑な処理をともなうコンポーネントの初期開発ではシステムエンジニアの支援を受けるとともに,頻繁に発生すると思われる機能変更などの保守についてはエンドユーザが単独でコンポーネントビルダを用いて行う方法が現実的と思われる.

たとえば，会議開催システムの会議室管理や備品管理のコンポーネントは，最初はシステムエンジニアがスケジュール表管理コンポーネントとして基本的なものを定義しておき，利用時にエンドユーザが必要に応じてカスタマイズできることが望ましい．

4. スクリプト言語

4.1 設計思想

スクリプト言語は，静的モデルの定義に用いるものであるが，本システムでは2つの役割を持つ．1つは，動的モデルからプログラムを自動生成してシミュレーション実行するときのプログラム記述言語としての利用である．業務コンポーネントが充実していれば，エンドユーザはスクリプト言語を意識する必要はない．しかし，アプリケーション開発時に不足しているコンポーネントはそれを新規開発する必要がある．そこで，スクリプト言語の2つ目の役割として，そのコンポーネント記述言語としての利用がある．

いずれの役割においても基本となるメッセージフローを自然な形で表現するために，メッセージセット記述機能を導入した．

4.2 メッセージセット

4.2.1 メッセージセットの役割

従来のオブジェクト指向言語では，送信側オブジェクトのメソッド内で受信側オブジェクトへの送信メッセージを記述する場合が多かった．この方法はオブジェクト間の協調関係を2者間に限定することで個々のオブジェクトの自律性を高めるという特徴を有する．しかしながら，この特徴は業務の専門家がワークフローシステムのような業務を分散協調型モデルとして表現する場合には必ずしも最善ではない．ワークフローは複数の業務担当者にまたがる業務の流れがその出発点で決まっている場合も多い．複数の回覧先を指定した書類の処理などがその典型である．

本システムでは，業務の専門家がメタファーベースのモデリングを行えるようにするために，従来のメッセージの概念を拡張し，複数のメッセージを組み合わせたメッセージセットの機能を導入した．すなわち，宛先のオブジェクト名とメソッド名をひと組みの単体メッセージとし，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オブジェクトどうしがやりとりする．

このように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メッセージセットを導入することにより，現実の業務手順と同じように，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できるようになる．したがって，ワークフローシステムにおいては全体のワークフローを管理するメタなシステムが不要になり，純粋な分散協調型モデルを構築できる．

4.2.2 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．

$M ::= M_s \| M_p \| X$

$M_s ::= [M, M, \ldots, M]$

$M_p ::= [M|M|\ldots|M]$

$X ::= obj.msg$

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

4.2.3 直列のメッセージセット

一般にオブジェクト指向言語ではメソッドの実行結果はメッセージの発信元に返される場合が多いが，アクタモデル[25]では結果の送り先を指定できる．M-baseのスクリプト言語ではさらに結果の送り先で実行されるメッセージも一緒に送ることができるようにした．

たとえば，簡単な例として，ある書類の審査業務において，A さんが査定して，B さんが審査して，C さんが承認するような直列のメッセージセットは，2番目の構文規則を用いて以下のように記述する．

$[a.assess, b.inspect, c.approve]$

オブジェクト a はこのメッセージを受け取ると assess メソッドを実行後，オブジェクト b に [b.inspect, c.approve] を送る．同様にオブジェクト b は inspect メソッドを実行後，オブジェクト c に [c.approve] を送る．

4.2.4 並列のメッセージセット

一方，たとえば A さん，B さん，C さんに会議開催通知を出す場合のように並列にメッセージを送る場合は，3番目の構文規則を用いて以下のように記述する．

$[a.announce|b.announce|c.announce]$

あるオブジェクトがこのメッセージセットを送信すると，各オブジェクトに announce メッセージが同時に送られる．

5. モデリング&シミュレーションツールと言語処理系の連携

5.1 スクリプトの生成

モデリング&シミュレーションツールで作成された動的モデルは，内部的には複合オブジェクトとして保存される．動的モデル全体に対応するオブジェクトの内部にエンドユーザがドラッグ&ドロップしたオブジェクト群のリストとそれらのオブジェクト間のメッセージ群のリストが保持される．シミュレーション実行時

にはこれらの情報からスクリプト言語のテキスト形式プログラムが自動生成される．

たとえば図3の動的モデルから次のようなスクリプトが自動生成される．

```
class Main {
    public main(String args[]){
        Class_事務局 事務局;
        Class_会議室管理 会議室管理;
        Class_備品管理 備品管理;
        Class_Staff 安部;
        Class_Staff 馬場;
        Class_Staff 千葉;
        事務局 = new Class_事務局 ();
        会議室管理 = new Class_会議室管理 ();
        備品管理 = new Class_備品管理 ();
        安部 = new Class_Staff();
        馬場 = new Class_Staff();
        千葉 = new Class_Staff();
            ⋮
    }
}
class Class_事務局 {
    public 会議開催 (Date d){
        [ 会議室管理．予約 (d), this.結果 (boolean),
            備品管理．予約 ()];
        [ 安部．通知 () | 馬場．通知 () | 千葉．通知 ()];
    }
}
```

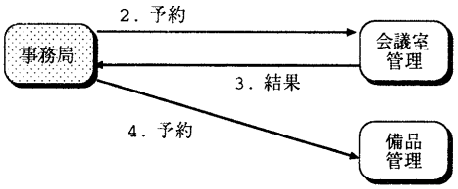
事務局オブジェクトの会議開催メソッドに注目すると，図3で示されたメッセージフローは，**図6** の (a) と (b) に示すように事務局オブジェクトを出発点とする2つの部分に分けられる．(a) はメッセージ番号の2，3，4に対応し，一筆書きで表現できる部分なので直列メッセージセットに変換されている．一方，(b) は同じメッセージ番号5に対応する3つのメッセージで，並列メッセージセットに変換されている．

5.2 スクリプトの実行

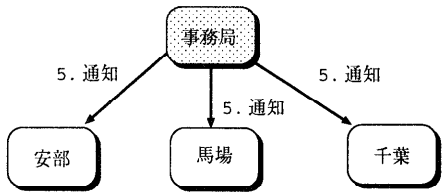
モデルのシミュレーション実行の容易性を考慮し，スクリプト言語はインタプリタで実行する方式とした．すなわち，上記の自動生成されたスクリプトはテキスト形式でスクリプト言語の処理系に渡され，構文木の中間形式に変換された後，解釈実行される．

実際のメッセージセットの実行手順を最初のメッセージセットの例で説明する．このメッセージセットは3つの単体メッセージを直列に実行する形式になっており，会議室管理の予約メソッドに渡される．そのメソッド実行中に **next**(...) 文に出会うと，その時点で残りのメッセージセット [**this**. 結果 (...), 備品管理．予約 ()] が呼ばれる．以下同様．

なお，各オブジェクトを分散環境下で実行する場合



(a) 直列メッセージセットの生成例



(b) 並列メッセージセットの生成例

図6 動的モデルからメッセージセットへの変換
Fig. 6 Examples of parts of dynamic model to be transformed into message sets.

は，プログラムの起動時に稼働させたいノード側でノード名，クラス名などを指定する．

6. 考　　察

6.1 ワークフロー

エンドユーザコンピューティングの分野では，ワークフローシステムのようなビジネス分野に共通の業務を対象にカスタマイズ可能なパッケージが普及し始めている．そこで，異なるワークフローシステム相互の間の接続性が重要であるという認識から，ワークフローシステムに関する標準化団体 WfMC（Workflow Management Coalition）やオブジェクト技術に関する標準化団体である OMG（Object Management Group）では，ワークフローシステムの参照モデルやインタフェースの標準化の努力がなされている．

M-base も主要な対象アプリケーションの1つとしてワークフローシステムを想定しているので，その観点で考察する．ただし，上記の標準化が主に部門間や企業間にまたがる比較的大規模のワークフローシステムを想定して，その相互運用性の実現を目指しているのに対して，本システムでは部門レベルの比較的小規模なシステムを想定しており，エンドユーザ自身が開発することを目的としている点が異なる．

まず，OMG の Workflow Management Facility に関する RFP [27] の中で規定されている5つの必須条件に関して本システムを評価する．第1のワークフローメタモデルについては，M-base では1オブジェクト1

業務というメタファーベースの分散協調型モデルを採用しており，エンドユーザに分かりやすくなっている．

第2のワークフロー実行規定については，M-baseでは，メッセージフロー図，メッセージセット，メッセージ変換という3種類のメッセージ表現を導入した．

メッセージフロー図は，モデリングツールで定義される．この方法はメタファーベースの業務モデル構築との対応がとりやすいので，エンドユーザには最も容易な方法と思われるが，複雑なロジックは記述できない．メッセージセットは，メッセージフローからの自動生成のほかに，スクリプト言語を用いて定義できるが，業務のワークフローに対応させることで，エンドユーザには容易な方法と思われる．

メッセージ変換はメッセージフロー図およびメッセージセットから導くことができる．たとえば図3の例では，事務局オブジェクトに関して次の2つのメッセージ変換を規定している．

会議開催 → 会議室管理．予約
結果 → 備品管理．予約 (),
[安部．通知 () | 馬場．通知 () | 千葉．通知 ()]

しかし，個々のワークフローに依存しない共通性の高いオブジェクトの仕様をスクリプト言語で定義した場合はメッセージ変換を直接定義したことになる．たとえば図3の会議室管理オブジェクトは，複数のワークフローシステムで共有される場合はあらかじめ次のメッセージ変換を有するコンポーネントとして与えられることになる．

予約 → this. 結果

本システムでは，業務モデルとの対応が可能な場合はメッセージフロー図あるいはメッセージセットを用いてワークフローを定義することを推奨する．この方式は，コンポーネントウェアの活用（再利用）という観点では，特定の業務向けアプリケーションフレームワークやデザインパターンの抽出につながる．

第3のワークフロー監視については，M-baseでは小規模なシステムを対象にしているので，ワークフロー管理のためのサブシステム[28]を持たないが，上記の3種類のメッセージ表現に対応するメッセージフロー制御方式として，集中制御と分散制御の2種類がある．メッセージフローの集中制御はメッセージセットあるいはそれを生成するメッセージフロー図によって達成され，実世界において業務のワークフローが出発点で決定される場合に対応する．メッセージフローの変更はメッセージセットあるいはそれを生成するメッセージフロー図の書き換えで済むので容易である．

一方，分散制御は個々のオブジェクトの入力メッセージと出力メッセージの関係を定義したメッセージ変換によって達成される．したがって，メッセージ変換の変更によってメッセージフローを変更する方法は，他のメッセージフローに意図しない変更をもたらす危険がある．

しかしながら，いずれの場合もM-baseでは一般のワークフローシステムでよく採用されているようなデータベースの集中管理はしていないので，ワークフローの状態の検索・問合せ機能は各オブジェクトの共通メソッドとして実装することになる．

第4のワークフロー実行履歴検査に関しては，システム実行系に実装すべき機能であるが，M-baseでは開発時のテストデバッグ機能に対応するシミュレーション機能を拡張する方式となる．

第5のワークフローのネストに関しては，大規模なワークフローシステムでは必須の機能と思われるが，M-baseではローカルな業務の自動化を意図しているので必要性は弱い．むしろ個々に開発された小規模のシステムがサブシステムとしてネットワーク間で接続されて大規模化されていくことを想定している．

一方，WfMCでは2つのワークフロープロセスの相互運用のための4種類の方式[29]を提示しているが，M-baseはメッセージ駆動の分散協調型モデルをベースにしているので，いずれのモデルへの拡張も可能である．また，WfMCではワークフローと個々のノードでのアプリケーション処理を明確に分けているが，M-baseでは両者は一体として扱い，個々のノードのオブジェクトが処理本体に対応する．さらにWfMCではエンドユーザとワークフローシステム管理者も明確に分離しているが，M-baseではエンドユーザ主導開発を目的としている．これらの違いはいずれも対象とするアプリケーションの規模の違いによるものである．

6.2 業務コンポーネント

最近，コンポーネントをベースにした新しいアプリケーション開発技法として，アプリケーションフレームワーク[30]やコンポーネントウェアが注目されている．その一例として，OMGでは，ビジネスオブジェクトと呼ばれる業務コンポーネントの標準化のためにBODTF（Business Object Domain Task Force）というタスクフォース[31]が設置されている．BODTFのビジネスオブジェクトもM-baseの業務コンポーネントもともにドメインモデルの中のオブジェクトとして位置付けられていることからほぼ同じ概念と考えられる．

一方，いくつかの相違点もある．BODTFが想定するビジネスアプリケーションアーキテクチャでは，ビ

ジネスエンティティオブジェクトとビジネスプロセスオブジェクトをビジネスオブジェクトとしているが，M-baseの業務コンポーネントは，動的モデルの中での利用を想定しているので後者に近い．ビジネスエンティティオブジェクトは業務コンポーネントの特殊な場合で，オブジェクト指向技術における抽象データ型に近い扱いをするオブジェクトが対応する．

BODTFでは，単一のビジネスオブジェクトが実装では大規模ネットワーク上で性能の最適化のために分割される可能性を示唆しているが，本研究では，先にも述べたようにむしろ個々のシステムがネットワーク間で接続されて大規模化されていくことを想定している．

7. おわりに

エンドユーザ主導のアプリケーション開発のための方法論とそれを支援する開発環境M-baseについて述べた．特に，業務モデルの動的な振舞いをメッセージフローとしてモデリングすることによりアプリケーションを構築するという方法を提案し，モデリング&シミュレーションツールとスクリプト言語が果たす役割とその実現方式を示した．

本システムの実装には，Java言語を用い，分散オブジェクトの通信制御にはJavaRMIを用いた．プログラムの大きさは，モデリング&シミュレーションツールは63クラスで約7,300行，スクリプト言語の処理系は55クラスで約6,000行である．

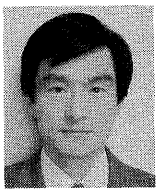
謝辞 本研究の一部はEAGL事業推進機構および（財）電気通信普及財団の支援を受けて実施したものである．

参考文献

1) Udell, J.: Componentware, *BYTE*, Vol.19, No.5, pp.46–56 (1994).
2) 青山幹雄，中所武司，向山 博（編）：コンポーネントウェア，共立出版 (1998).
3) Johnson, R.E.: Frameworks = (Components + Patterns), *Comm. ACM*, Vol.40, No.10, pp.39–42 (1997).
4) 日本情報処理開発協会：エンドユーザ向けアプリケーション統合環境の研究開発報告 (1996, 1997).
5) 吉田武稔，志村 功，田中一郎：ビジネスオブジェクトとその実現のための基盤技術，情報処理，Vol.39, No.2, pp.81–85 (1998).
6) Monarchi, D.E. and Puhr, G.I.: A Research Typology for Object-Oriented Analysis and Design, *Comm. ACM*, Vol.35, No.9, pp.35–47 (1992).
7) Fichman, R.G. and Kemerer, C.F.: Object-Oriented and Conventional Analysis and Design Methodologies, *IEEE Computer*, Vol.25, No.10, pp.22–39 (1992).
8) Shlaer, S. and Mellor, S.J.: *Object-Oriented Systems Analysis*, Prentice-Hall (1988).
9) Wirfs-Brock, R., Wilkerson, B. and Wiener, L.: *Designing Object-Oriented Software*, Prentice Hall (1990).
10) Booch, G.: *Object-Oriented Design with Applications*, Benjamin/Cummings (1991).
11) Coad, P. and Yourdon, E.: *Object-Oriented Design*, Prentice-Hall (1991).
12) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-Oriented Modeling and Design*, Prentice-Hall (1991).
13) 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，No.95-SE-104-4 (1995).
14) Chusho, T., Konishi, Y. and Yoshioka, M.: M-base: An Application Development Environment for End-users Computing based on MessageFlow, *Proc. APSEC '96*, pp.366–375, IEEE Computer Society (1996).
15) 松本光由，小西裕治，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開発環境M-baseにおける分析・設計技法，情報処理学会オブジェクト指向 '97 シンポジウム，pp.128–135 (1997).
16) 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，pp.87–94 (1996).
17) Grudin, J.: Computer-Supported Cooperative Work, *IEEE Computer*, Vol.27, No.5, pp.19–26 (1994).
18) Clements, P.C. and Northrop, L.N.: Software Architecture: An Executive Overview, *Component-Based Software Engineering*, pp. 55–68, IEEE CS (1996).
19) 岩田智彰，中所武司：分散アプリケーションソフトウエア開発環境M-baseにおけるメッセージフロー制御方式，情報処理学会第55回大会講演論文集（1），pp.424–425 (1997).
20) Jacobson, I., et al.: *Object-Oriented Software Engineering*, Addison-Wesley (1992).
21) Mellor, S.J. and Johnson, R.: Why Explore Object Methods, Patterns, and Architectures?, *IEEE Software*, Vol.14, No.1, pp.27–30 (1997).
22) Tepfenhart, W.M. and Cusick, J.J.: A Unified Object Topology, *IEEE Software*, Vol.14, No.1, pp.31–35 (1997).
23) Fayad, M. and Schmidt, D.C. (Eds.): Object-

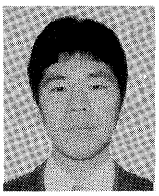
Oriented Application Frameworks, *Comm. ACM*, Vol.39, No.10, pp.32–87 (1997).
24) Gamma, G., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns*, Addison-Wesley (1995).
25) Hewitt, C.: Viewing Control Structures as Patterns of Passing Messages, *Journal of Artificial Intelligence*, Vol.8, No.3, pp.323–364 (1977).
26) 小西裕治：分散協調型アプリケーションのためのオブジェクト分析・設計言語 Hoop の開発とその評価，明治大学理工学研究科 1996 年度修士論文 (1997).
27) Workflow Management Facility Request For Proposal, OMG Document: cf/97-05-06 (1997).
28) Wallnau, K., Long, F. and Earl, A.: Toward a Distributed, Mediated Architecture for Workflow Management, *Component-Based Software Engineering*, pp.69–74, IEEE CS (1996).
29) Hollingsworth, D.: The Workflow Reference Model, WfMC, Doc. No.TC00-1003 (1994).
30) IBM 資料（翻訳版）：サンフランシスコ・プロジェクト技術要約, http://www.developer.ibm.com:8080/welcome/java/sftecovr.html (1997).
31) Casanave, C.: Business-Object Architectures and Standards, OMG BODTF, http://www.dataaccess.com/bodtf/BOPaper.htm.

(平成 10 年 4 月 22 日受付)
(平成 11 年 2 月 8 日採録)



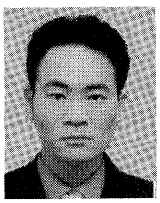
中所 武司（正会員）

1946 年生．1969 年東京大学工学部電子工学科卒業．1971 年同大学院修士課程修了．同年（株）日立製作所入社．同社システム開発研究所主任研究員を経て，1993 年から明治大学理工学部情報科学科教授，現在に至る．ソフトウェア工学の研究に従事．エンドユーザ主導のアプリケーション開発方法論に関心を持つ．工学博士（東京大学）．1982 年度情報処理学会論文賞，1986 年度大河内記念技術賞受賞．著書「ソフトウェア工学」（朝倉書店），「ソフトウェア危機とプログラミングパラダイム」（啓学出版），「プログラミングツール」，「人工知能」（昭晃堂，共著）．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，IEEE Computer Society，ACM 各会員．



小西 裕治

1971 年生．1995 年明治大学理工学部情報科学科卒業．1997 年同大学院修士課程修了．同年日立ソフトウェアエンジニアリング（株）入社．現在携帯端末のソフトウェア開発の業務に従事．



松本 光由

1973 年生．1996 年明治大学理工学部情報科学科卒業．1998 年同大学院修士課程修了．同年（株）日立製作所入社．現在同社情報システム事業部にて，官公庁関連の大規模システムの開発業務に従事．オブジェクト指向分析・設計技法，アプリケーションフレームワークに関心を持つ．