ソフトウェア工学 120－18
（1998．7．24）

「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におけるコンポーネントベース開発技法

岩田智彰† 中所武司†

近年，パソコンの普及と共に，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．そのためには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，業務モデルと計算モデルの一致という基本コンセプトを設定した．我々は，このコンセプトを達成するためにオブジェクト指向概念に基づくアプリケーション開発環境 M-base の実現を目指している．アプリケーションの開発はコンポーネントベースのビジュアルツールを用いて行う．ユーザインタフェースをモデルから半自動的に生成することやメソッドの定義にもコンポーネントを利用することなどにより，エンドユーザ主導型のアプリケーション開発技法を実現した．

Component-Based Application Development Environment, M-base, on the concept of "a Domain model ≡ a Computation model"

TOMOAKI IWATA† and TAKESHI CHUSHO †

Explosive increase in enduser computing on distributed systems requires that endusers develop application software by themselves. One solution is given as a formula of "a domain model ≡ a computation model." This formula implies that one task in a domain model of cooperative work corresponds to one object in a computation model based on an object-oriented model. Application development environment, **M-base**, supports this formula for cooperative systems such as groupware and work flow systems. Application is developed with a visual tool based on components. User interface is made semi-automatically from the model. Endusers complete it while adding components. Enduser-initiative application development techniques are achieved.

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[1)2)]．

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**[3)~5)] を開発中である．エンドユーザコンピューティングの分野を対象としているため，**M-base** による開発では，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挙げるプロトタイピングアプローチをとる．

コンポーネントベースのビジュアル開発ツールとしてモデリング&シミュレーションツールとユーザインタフェース (UI) ビルダーを試作した．またそれらから作成されたアプリケーションを分散環境下で動作させるための実行環境を試作した．

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章では開発環境の概

† 明治大学大学院理工学研究科基礎理工学専攻情報科学系
Graduate School of Science and Technology, Major in Sciences, Computer Science Cource, Meiji University

☆ 本研究の一部は EAGL 事業推進機構の支援を受けて実施したものである．

要，5章ではモデリング，6章ではユーザインタフェース (UI)，7章ではシミュレーション，8章では実行の制御について述べる．

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，本研究では，ユーザ企業の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を対象とする．特に，後者の人達は，業務の専門家として，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しており，今後急速に増加すると思われる．

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 [6] 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が重要である．

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，まずシステムエンジニアが基本となるコンポーネントを用意しエンドユーザはそれを使用して開発を行うが，その後のシステムやコンポーネントの保守・拡張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に分散オブジェクト指向設計技法を適用するためには，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振舞いを記述することが重要である．我々は，次に示すようなマクロモデルとミクロモデルの2階層モデルを導入して，基本的コンセプトを実現した．

(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

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インスタンスベースのドメインモデルを以下のように OAM (Object-based 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

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

と表現したメッセージ変換の集合である．また，ドメインモデル OAM の外界を仮想的にオブジェクトとみなして，外界からドメインモデルへのメッセージ集合 Min，ドメインモデルから外界へのメッセージ集合 Mout を決定する．

3.3 設計モデルの構築手順

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

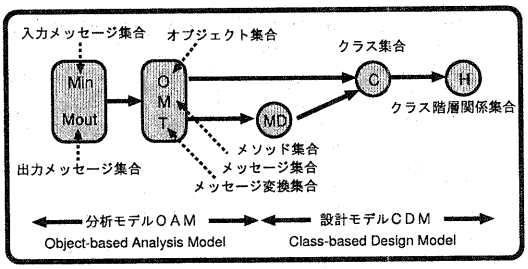

図 1 M-base のモデリングプロセス

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．

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
- コンポーネントビルダ
- 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（設計モデル）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．このようなコンポーネント化を支援するために

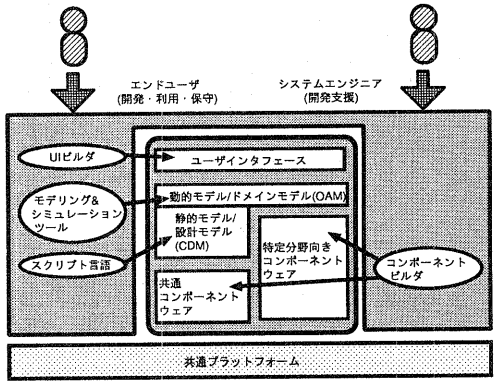

図 2 M-base の開発環境（外側）とアプリケーション・アーキテクチャ（内側）

コンポーネントビルダを用意する．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．

5. モデリング

5.1 モデリング手順

メッセージフローが本質的な意味を持つ分散型アプリケーションソフトウエアは計算モデルもデータモデルも確立していないことがことが多い．そこで，オブジェクト間の静的な関係に先立ってメッセージの送受信によって規定されるオブジェクト間の動的な関係，すなわち，システムの振る舞いを定義する必要がある．

そのため **M-base** ではモデリングツールを使用することでシステムの動的な振る舞いをコンポーネントベースでビジュアルに定義することを可能とした．具体的には各業務処理を表すコンポーネントを矢印で結び付けていくことで簡単に業務プロセスを定義でき，修正も簡単に行える．

アプリケーションを作成する手順は以下の通りである．

(1) オブジェクトとメッセージを抽出する
(2) ドメインモデルを定義する
(3) 各コンポーネントの詳細を定義する
(4) UI を作成する
(5) シミュレーションを行い妥当性の検証をする

次のような会議室開催業務を例題として開発手順を説明する．オフィスで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 OHP などの備品の予約を行うと共に，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い出欠の管理を行う（図 3）．この業務を自動化するアプリ

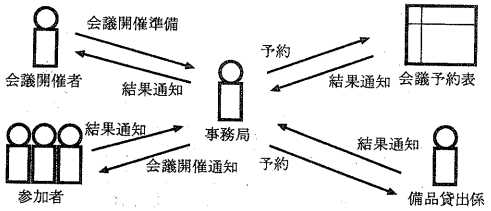

図 3　会議開催業務の概要

ケーションの開発を想定する．

5.2　オブジェクトとメッセージの抽出

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．

例）　会議開催システムの場合，オブジェクトは会議開催者，事務局，会議室，備品，参加者の5つとなる．

- 開催者
  会議開催準備の要求を事務局に送る．
- 事務局
  会議開催準備の要求メッセージを受け取ると，会議室の予約，備品の予約，その会議のメンバへの会議開催通知など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って管理する．
- 会議室
  会議室予約の要求メッセージを受け取って予約管理をする．
- 備品
  備品予約の要求メッセージを受け取って予約管理をする．

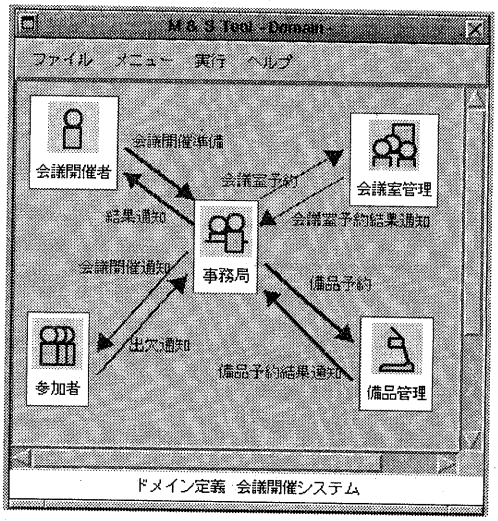

図 4　ドメインモデルの定義

表 1　オブジェクトの定義例

オブジェクトの名前	事務局
入力メッセージ（メソッド名）	会議開催準備，会議室予約結果通知 備品予約結果通知，出欠通知
出力メッセージ（イベント名）	会議室予約，備品予約 会議開催通知，結果通知

- 参加者
  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事務局へ返送する．

5.3　ドメインモデルの定義

抽出したオブジェクトとメッセージをドメインモデルにマッピングする．具体的にはモデリングツールを使い，オブジェクトをコンポーネントとして定義し，そのコンポーネント間に矢印を引くことでメッセージを定義する．

例）　会議開催準備に関するドメインモデルの定義は図4のようになる．

このコンポーネントは，ユーザが定義する方法とあらかじめ用意されているコンポーネントを使用する方法の2種類がある．

ユーザが定義する場合，業務の専門家はウィザードにしたがって以下のような入力を行うことによりコンポーネントの定義を簡単に行うことができる．

(1)　オブジェクトの名前の定義

(2)　入力メッセージの定義

(3)　出力メッセージの定義

例）　事務局のオブジェクトの定義は表1のようになる．

一方，M-base であらかじめ用意されているもの

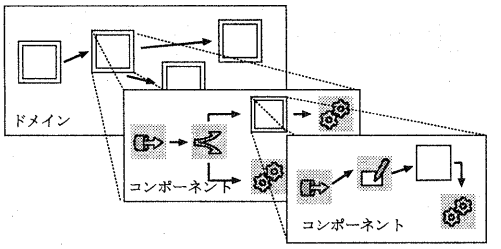

図 5　コンポーネントの定義

を特定業務向けコンポーネントという．これはエンドユーザの負担を減らすために特定業務領域に限定してあらかじめ定義されているものである．

ユーザ定義で業務の専門家が定義したコンポーネントを特定業務向けコンポーネントとして登録することも可能とした．これにより今後作成するアプリケーションで再利用をすることが可能となる．

5.4　コンポーネント定義

5.4.1　定義方法

M-base では，再利用性を考慮して JavaBeans[7] のコンポーネントモデルに準拠したコンポーネントをエンドユーザが作成する方法を提供する．

定義方法は始めにコンポーネントを配置する．ここで配置するコンポーネントには以下のようなものがある．

- 制御用コンポーネント
  仕事の流れを制御する部品
- 共通コンポーネント
  汎用的に利用できる部品
- 特定業務向けコンポーネント
  特定業務領域に限定した部品

その後これらコンポーネント同士を繋ぎ合わせ，以下の情報を記述することで簡単にドメインモデルのコンポーネントを定義することを可能とした．図 5 は，ドメインモデルを構成するコンポーネントの 1 つを，2 階層で定義している例である．

- メッセージの元，先のオブジェクトを指定する
- メッセージの名前，説明を記述
- メッセージが送信される理由（イベント）の選択
- メッセージが実行する処理の選択
- メッセージを実行するのに必要な情報の対応づけ（引数の設定）

5.4.2　制御用コンポーネント

制御用コンポーネントは仕事の流れの制御，流れているデータの変更を行うためのものである．制御コンポーネントには以下のようなものがある（図 6）．

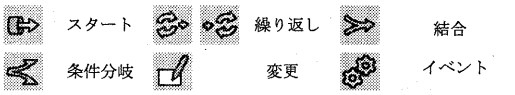

図 6　制御用コンポーネント

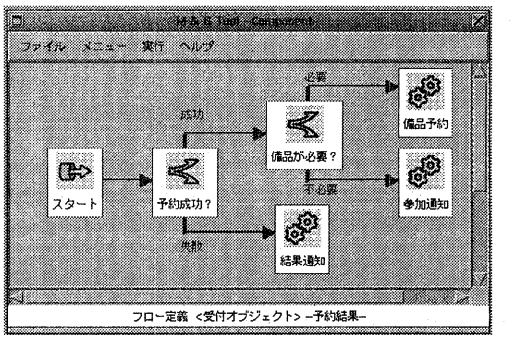

図 7　会議室予約結果通知メソッド定義

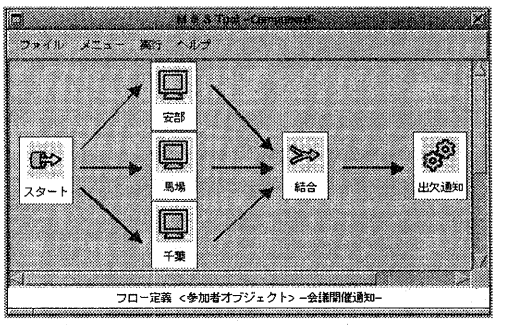

図 8　会議開催通知メソッド定義

- スタート
  処理の開始を示すものである．
- 条件分岐
  入力情報に依存した条件によって異なる出力処理を行うものである．
- 繰り返し
  回数を指定した繰り返しと，条件が真の間繰り返す方法の 2 種類がある．
- 結合
  複数のメッセージフローの合流点を示す．
- 変更
  これは通過する際にメッセージの値を変更するものである．
- イベント
  他のメソッドを起動するメッセージの発生を示す．

例 1）　会議開催システムで，事務局が会議室に予約を申し込み予約結果が戻ってきた際の処理（会議

室予約結果通知メソッド）の定義を図7に示す．この処理から発生するメッセージは以下の3つである．

- 予約が失敗した場合，会議開催者に知らせる「結果通知」
- 予約が成功し備品が必要な場合，備品の予約を行う「備品予約」
- 予約が成功し備品は必要でない場合，参加者に開催を通知する「参加通知」

そこでそれぞれのイベント発生のコンポーネントを配置する．次に，それぞれのイベントが起る条件を設定する．そのために，予約が成功かどうか，また備品が必要かどうかというの2種類の条件分岐を付け加える．最後にそれらコンポーネント間のメッセージの流れを設定する．

例2） 参加者に対して会議の開催を通知する処理（会議開催通知メソッド）の定義を図8に示す．これは安部さん，馬場さん，千葉さんがUIコンポーネントで参加登録を行い，結合コンポーネントにより3つの結果が集まったら事務局に出欠通知を返すことを示す．

5.4.3 共通コンポーネント

これは汎用的な処理を行うコンポーネントのことを指す．具体的な例としては，ユーザインタフェース，ファイル操作，データベース処理，メール送信，プリンタ印刷などがある．

6. UIの定義

アプリケーションのUIはコンポーネントとして定義する．UIビルダーとの連携によって次のような方法でUIを作成する．

(1) モデリングツールでの情報の定義
(2) 画面の自動生成
(3) 画面のカスタマイズ

以下では，これらについて説明をする．

6.1 モデリングツールでの情報の定義

モデリングツールでUIコンポーネントに必要な以下の情報をウィザードに従い定義する．

- 入力情報
  利用者が画面から入力する情報の名前と説明を定義する．
- 入力情報の種類
  情報のタイプを定義する．例えば，データが文字型・数値型などのうちどれか．入力は必須であるのか．数値型ならば入力可能な範囲などを定義する．
- 出力情報

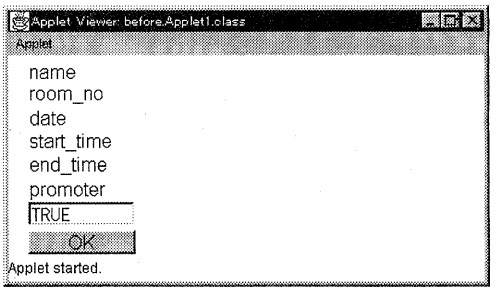

図9 修正前のUI

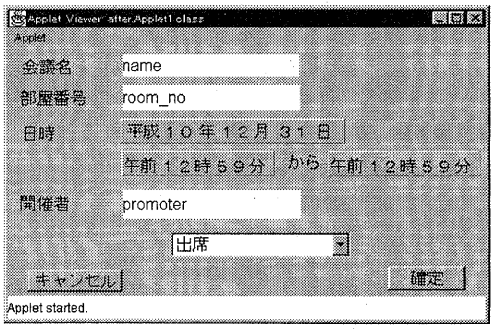

図10 修正後のUI

最初の画面表示時に出力する情報と，何らかのアクションを起こした後に結果として返ってくる出力情報を定義する．

例） 参加者コンポーネントの会議開催通知メソッド（図8）の中で会議参加者が出欠の入力に使用するユーザインターフェース（参加登録UI）を作成する場合，出力情報は画面表示時に必要な会議名，部屋番号，開催日，開始時間，終了時間，開催者である．また入力情報は会議に出席するか欠席するかである．そこで，種類はboolean型となり入力は必須である．

6.2 画面の自動生成

ドメインモデルからの情報の種類に基づき，UIビルダーはUIにコンポーネントを自動的に配置する．例えば，文字を入力する項目ならテキストボックス，出力する項目ならラベルが自動的に配置される．

例） 参加登録UIでは，画面上に出力用に会議名，部屋番号，開催日，開始時間，終了時間，開催者が配置される．また，入力用にテキストボックスが配置され，入力を確定させるためのボタンが一つ配置される．図9は自動的に配置された状態である．

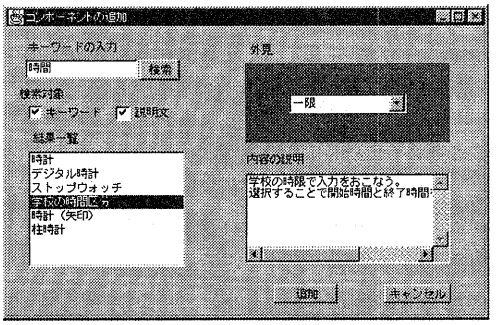

図 11　特殊コンポーネントの検索

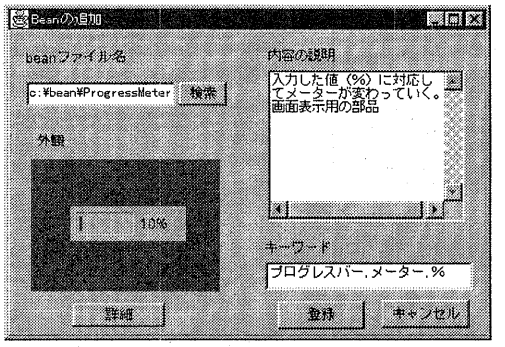

図 12　JavaBeans の追加用ツール

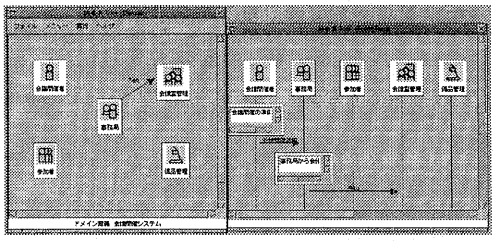

図 13　シミュレーションツール

6.3　画面のカスタマイズ

画面のカスタマイズ機能は大きくわけて以下の 2 種類である．

- 自動配置された部品に対して修正
- 新規 GUI コンポーネントの追加

修正というのは，具体的には，コンポーネントの位置，大きさ，色を変えたり，デフォルトで表示される文字を定義する．

一方追加では以下の 2 種類の GUI コンポーネントを追加することが可能である．

- 基本コンポーネント
- 特殊コンポーネント

基本コンポーネントというのは，ボタンやテキストボックスなど基本的な機能のみを持った GUI コンポーネントである．

またもう一方の特殊コンポーネントというのは，良く使われる複合的な機能を持った GUI コンポーネントである．例えば，カレンダーが表示されていて，その日付の欄を選択した場合，年，月，日の 3 つの値を得ることができるコンポーネントなどである．

例）参加登録 UI として半自動的に作成された画面では出力される情報が何を意味するのかが利用者には理解が困難である．そこでそれぞれにタイトル用のラベルを付加し位置を修正する．次に，出席・欠席を入力するために True,False をテキストで入力するのは不便なので，リストボックスを付加して出席か欠席を選択させる．それをテキストと関連づけてリストボックスを選択した時点でテキストの内容を変更するように修正し，テキストボックスは非表示とする．また，日付と時間の表示をただテキストで表示させる代りに特殊部品を付加して見易くする．その結果，図 10 のようになる．

これら GUI 部品が増えてきた場合，目的にあったものを探すことが困難になると考えられる．そこで，部品を追加する際にキーワードを入力することにより候補となる部品，外観，説明が表示される機能を付加した．図 11 は，「時間」というキーワードから「学校の時間区分」を選択している例である．

また GUI コンポーネントは JavaBeans で作成されているので，ネットワーク上で配布されているコンポーネントなども簡単に登録でき特殊コンポーネントとして使用可能である．図 12 はネットワーク上からダウンロードしてきたプログレスバーをツールを使用して特殊コンポーネントとして登録している画面である．

7.　シミュレーション

シミュレーションツールはモデリングツールにより作成した業務の流れの妥当性をビジュアルにチェックするためのものである．

この際シミュレーションを行う方法としては以下の 2 種類がある．

- オブジェクト間のメッセージパッシング表示
- イベントトレース

前者はオブジェクト間で実際にメッセージが流れている部分が矢印となって表示され，現在行われている処理がわかる（図 13 左）．この方法では，作成したモデル上での動作の確認を行うことができるので直感的

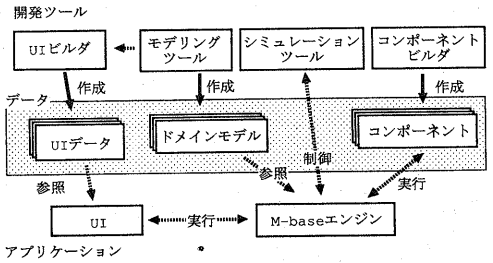

図 14 実行制御方式

に認識がしやすい．しかし，これでは現時点でのメッセージの流れしかわからないため，過去の事象や原因となるものが分かりにくいという点がある．

一方後者では，一つのオブジェクトに対して一つの区画が与えられすべてのイベントが事象トレース上に表されるので全体のシーケンスを見渡しやすい（図13右）．

8. 実行の制御

M-base ではアプリケーションは利用開始後も頻繁に変更拡張が行われることや各コンポーネントはネットワーク上に分散されていることなどを考慮して図14のような実行制御方式[5]とした．

M-base エンジンは次のような手順でアプリケーションの実行を行う．

(1) UIから入力データを受け取る．
(2) 次に実行すべきメッセージを決定する．
(3) 実行対象となるコンポーネントをネットワーク上から探しだし実行する．
(4) 終了まで (2), (3) を繰り返す．

これにより次のような利点が生まれる．

- モデリングツールでドメインモデルを変更するだけで動作の変更を行うことができる．
- ネットワーク上に分散しているコンポーネントを透過的に実行することができる．

9. 結果の検討

M-base が対象としているような中・小規模のシステムでは，変更・拡張が頻繁におこり動的な振る舞いが複雑なものが多い．しかし，既存のワークフローソフト製品はフローに基づくモデリングを行うためフローの変更が起った場合修正部分が多くなると考えられる．それに対し，**M-base** ではオブジェクト指向に基づく分散協調型問題解決モデルを用いているので，修正個所がわかりやすく容易である．

また，モデルの変更・修正に伴いビューであるユーザインタフェースの修正も頻繁に起る．ところが，もともとユーザインタフェースの作成はアプリケーション作成の中でもかなりの時間を占めるものである[8]．そこで，UIの半自動生成機能により，モデルが確定するまでは半自動的に作成されたUIを使用してシミュレーションを繰り返し，確定してからUIを改良することを可能とした．これによりエンドユーザの負担が軽減できたと考える．

10. おわりに

本稿では，コンポーネントベースの開発技術を用いることによりエンドユーザ主導のアプリケーション開発が可能であることを述べた．しかし，エンドユーザだけで保守を行うためには，さらに管理に関する支援機能が重要になる[9]．そこで今後はドメインモデルの変更履歴の管理やドキュメントについて研究を行う予定である．また，エンドユーザによるアプリケーション開発を通して評価をおこなっていきたい．

参考文献

1) 青山幹雄: コンポーネントウェア：部品組み立て型ソフトウェア開発技法，情報処理学会誌 Vol.37, No.1, pp.71-79, (1996).
2) 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書，日本情報処理開発協会 (1997).
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).
4) 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** における分析・設計技法，情報処理学会 OO'97 シンポジウム pp.128-135(1997).
5) 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」，第55回全国大会講演論文集 Vol.1, No5AE-7 pp.425-426(1997).
6) 堀内正博，飯島淳一：「ワークフロー管理システムの有効性について」，オペレーションズ・リサーチ，Vol.41 No.10 pp.559-568 (1996).
7) Graham Hamilton: The JavaBeans API specification, Sun Microsystems, http://www.javasoft.com/beans/docs/beans.101.ps, (1997).
8) ビジュアルインタフェースの研究開発報告書，日本情報処理開発協会,pp.21-40 (1997).
9) Daniel Tkach, Walter Fang, Andrew So: Visual Modeling Technique : Object Technology Useing Visual Programming, Addison-Wesley (1996).