メッセージフローモデルに基づく エンドユーザ主導型アプリケーション構築・検証技法

石 樽 久 嗣[†] 紺 田 直 幸[††] 中 所 武 司[†]

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，開発の中心としてクラス図を使用する方法はエンドユーザには難しいので，インスタンスベースのモデル構築を基本とした．エンドユーザはコンポーネントベースの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を構築し，シミュレーション実行により動作確認を行う．業務モデルにおけるフロー分岐の記述には，従来のスクリプト言語や制御コンポーネントに代わりルールを用いることにより，簡素で柔軟なワークフローを実現した．またユーザインタフェースはモデルから自動生成することでユーザの負担を軽減し，さらに UI 遷移図を表示することにより，エンドユーザの要求を早期に洗練・検証することを可能にした．

Enduser-Initiative Application Development and Verification Based on Message Flow Model

Hisashi Ishigure,[†] Naoyuki Konda[††] and Takeshi Chusho[†]

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. Communication among objects is performed by rules instead of a script language or control components for implementation of flexible work flow. User-interface is generated automatically, and may be customized if necessary. Because transition diagram of UI are displayed automatically, it is easy for end-users to understand application-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，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ツールが実用化されはじめている[1)]．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．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業務に限られている．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 年代前半から多くのオブジェクト指向分析・設計技法が提唱されてきた[2)~4)]．OMG により標準として認められた UML[5)] によりモデルの表記法は統一されたが，プロセスは明記されていないため，その後も UML を用

† 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, ComputerScience Cource, Meiji University
†† 現在，ソニー（株）勤務
Sony, Co. Ltd.

いた方法論が数多く出現している．最近ではコンポーネントベースの開発手法として Catalysis[6] が注目を集めている．Catalysis では，オブジェクトの抽象レベルをタイプと呼ぶ．このタイプのインタフェースを先に決定し，その後内部を詳細化することによりモデル化を行う．またタイプモデルを作成する際，設計レベルのフレームワークであるモデルフレームワークを適用することにより，モデルの再利用を行える．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．

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を設定した[7]．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** を実現した[8)~11)]．これまでに **M-base** 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[8]，オブジェクト間の交信をメッセージセット単位で行うスクリプト言語[8]，ユーザインタフェースの自動生成を行う UI ビルダ[9] などを開発した．しかしながら，スクリプト言語を用いたフロー分岐記述の限界，エンドユーザによる要求分析の曖昧さ，シミュレーションツールにおけるモデル検証の限界などが課題とされてきた．

本稿では，**M-base** の概要を説明した後，上記の問題を解決する方法として，ルールを用いたフロー分岐記述[10]，UI 遷移図を用いた要求の洗練・検証[11]，プロパティシートを用いた厳密なシミュレーション検証について述べる．

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，市販のパッケージソフト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．

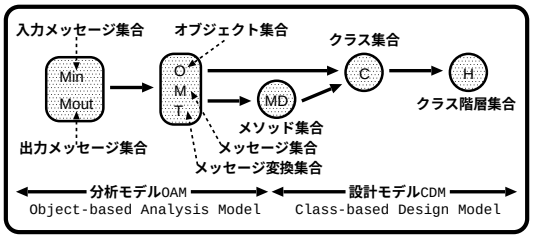

図 1 **M-base** のモデリングプロセス

2.2 モデリングの基本プロセス

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振舞いを明確にした後，必要に応じて静的構造を定義するという手順をとる．モデリングの初期段階で，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．

このモデリングプロセスの形式化は文献[7]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．そのために，開発対象のアプリケーションのソフトウェア・アーキテクチャは，図 2 の内側に示すように，ユーザインタフェース，モデル，コンポーネントウェアの 3 つの部分から構成される．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，3 層アーキテクチャの適用，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．

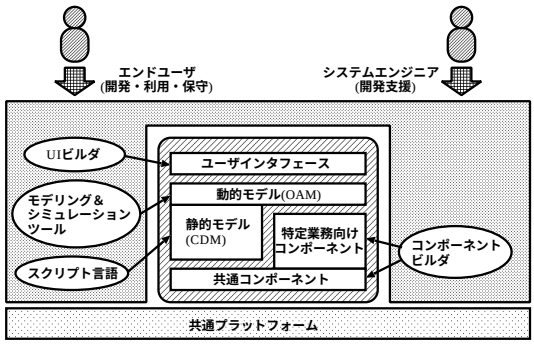

図 2 M-base の開発環境（外側）とアプリケーション・アーキテクチャ（内側）

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．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．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ある．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．

2.5 エンドユーザのモデリングプロセス

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構築である．動的モデルの構築は，業務コンポーネントの組合せが基本となるが，そこで導入したコンポーネントのうち既存のもの以外は，ユーザが新たに定義する必要がある．そこで M-base では，先に定義した動的モデルから静的モデルのスケルトンを自動生成することにより，エンドユーザによる新規オブジェクトの定義を容易にした．

M-base におけるアプリケーション開発は，次の手順で行われる．

(1) 業務仕様の詳細化
(2) ドメインモデルの作成
(3) ユーザインタフェースの構築
(4) シミュレーション実行による検証

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．

3. 適用事例

3.1 例題の説明

ここでは例題として，情報処理学会ソフトウェア工学研究会要求工学ワーキンググループにおいて共通問題とされている「国際会議のプログラム委員長の業務」[12] を取り上げ，実際の開発手順に従って M-base のアプリケーション開発技法を説明する．

3.2 業務仕様の詳細化

まず，対象とする業務の仕様を明確にするためにシステムの利用者を抽出し，そこから開始される機能概要（ユースケース[2]）を記述する．例題でのシステムの利用者はプログラム委員長であり，業務仕様に記述された 11 項目の業務のうち，前半の５項目に注目して，次のような４つのユースケースを抽出して業務の詳細化を行った．

- **スケジュールの決定**
  開催日を入力することにより，過去の同様の国際会議の経験的データから原案を自動的に生成し，これを修正してスケジュールの決定を行う．
- **CFP の作成・配布**
  原本となるホームページ掲載版は，表示の雛形を用意し，各項目に順に入力することにより作成する．次に，これを用いてメーリングリスト利用版とポスター版を自動生成する．
- **プログラム委員選出**
  プログラム委員候補名簿とプログラム委員名簿作成のためのデータは，手作業で入力する．プログラム委員就任依頼は電子メールで行い，プログラム委員が決定したら，名簿からメーリングリストを自動生成する．
- **投稿論文登録**
  投稿論文に論文番号を割り当て，投稿論文を手作業で登録し，投稿論文名簿を作成する．その後，投稿者に論文を受け取った旨の確認メッセージを電子メールで自動的に出す．

3.3 ドメインモデルの作成

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．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により，オブジェクトやメッセージの抽出を行う．このような動的モデルの構築はユースケース単位で行う．１つのシステムを複数のユースケースに分割することにより，モデリングの複雑さを回避できる．

国際会議のプログラム委員長業務のドメインモデル

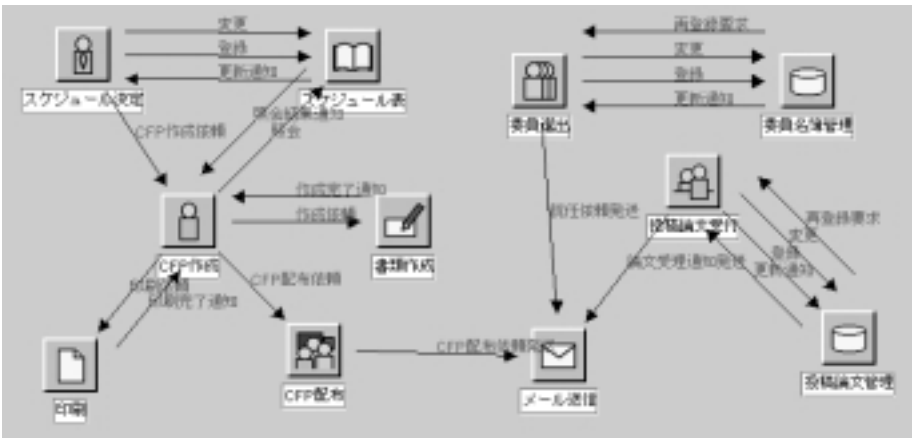

図 3 国際会議のプログラム委員長業務支援システムのドメインモデル

を図 3 に示す．抽出された業務オブジェクトは 11 種類で，オブジェクト間のメッセージも決定している．

ここで，あらかじめ適切なコンポーネントが用意されていない場合は，動的モデルからコンポーネントのスケルトンを自動生成する．具体的には，メッセージ属性（パラメータ）をプロパティに，メッセージ名をメソッドにそれぞれ割り当てることにより，静的構造のスケルトンを生成する．その後，生成されたメソッドに対して業務の用語を用いて処理内容を詳細化し，必要に応じてスクリプティングへと移っていく．この機能の実現により，動的モデルから静的モデルへの変換をシームレスに行うことができ，エンドユーザの負担を軽減できる．

3.4 ユーザインタフェースの構築

M-base における開発では，エンドユーザの負担を軽減するために，ユーザインタフェースはシステムから自動生成する．エンドユーザは必要に応じてこの UI をカスタマイズすることで，使い勝手の向上を図ることが可能である．すなわち，アプリケーション開発中のデバッグ作業では，UI 作成に手間がかからないように自動生成された UI を用い，完成後は利用者が使いやすいようにカスタマイズを行うことを想定している．

3.5 シミュレーション実行による検証

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．シミュレーションもユースケース単位で行う．図 4 はシミュレーションの実行画面であり，スケジュール決定オブジェクトがスケジュールを決定した後，CFP 作成オブジェクトに CFP の作成を依頼しているところである．このように業務フローの各メッセージをステップ実行し，起動したメソッドの内容をひとつずつビジュアルにチェックできることにより，エンドユーザによるシミュレーション実行を容易にしている．

図 4 シミュレーション実行画面

以上，**M-base** の概要について述べてきたが，以下では，主要な技術課題とその解決法について述べる．

4. フロー分岐の記述方法

4.1 ルールを用いた分岐記述

アプリケーションの実行時において，あるメソッドを実行した後，次にどのオブジェクトへメッセージを送信すべきかを動的に決定しなければならない場合がある．これまでに **M-base** では，スクリプト言語を用いる方法 [8] と，制御コンポーネントによりビジュアルに行う方法 [9] を実現してきた．

スクリプト言語では，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行う．このメッセージセットにガードと呼ぶ条件を付加することにより，メッセージの流れを動的に変化させることができる．しかしながら，ガードを適切に記述するにはある程度のスキルが要求されるため，これをエンドユーザ自身が行うのは難しかった．

一方，制御コンポーネントを用いてビジュアルに行う方法では，条件分岐，繰り返し，結合などの制御のためのコンポーネントを用意して，これらを結線することでフローを表現するため，エンドユーザ向きの方法と思われた．しかしながら実際には，複雑な処理を表現する場合には図が見辛くなり，全体を見通すことが困難になるため，修正を行う場合にユーザへの負担が大きくなるなどの欠点があった．

そこでこれらの問題を解決するために，柔軟性や記述容易性を考慮してルールを用いてメッセージフローの分岐を記述することにした．ルールはオブジェクトの各メソッドに対応させて記述する．オブジェクトごとに記述することも可能であるが，複雑な振る舞いをするオブジェクトの場合にルールの管理が困難になることや，ルールは一つのメソッドにのみ依存する場合が多いことからメソッドに対応させることにした．ルールは条件部と行動部から構成される．条件部はプロパティの値による論理演算の組合せにより表現し，行動部は次に実行するメッセージ名を指定する．例えば，CFP 作成オブジェクトが作成完了通知を受けるときのメソッド（書類受理メソッド）におけるルールは次のように記述できる．

（1） もし 作業継続=true && 印刷の有無=true ならば 印刷依頼 を行う
（2） もし 作業継続=true && 印刷の有無=false ならば CFP 配布依頼 を行う
（3） もし 作業継続=false ならば 作業終了 を行う

実行時のルールの選択方法については，エンドユーザの記述容易性を考慮して，メソッド実行後に対応するルールを上から順に評価を行い，最初にマッチしたものを選択して次のメッセージを送信するという単純な方式を採用した．

ルールは，構造が単純でありかつ柔軟に対応できるため，エンドユーザによる多種多様な業務のモデリングに有効な手段であるといえる．そのため，一般のワークフローソフト [13] でも，ルールを用いてフローを制御するものがある．これらは主に，金額による分岐や書類の承認・不承認のための分岐など，業務プロセスの一部としてルールを記述する場合に限られている．**M-base** においてもこのような使い方は可能である．さらに **M-base** では，オブジェクトのメソッド単位でルールを記述することにより，オブジェクト単位の保守や，ルールもオブジェクトに含めて再利用することが可能となった．また，各オブジェクトがルールに従って振る舞っていると捉えることにより，局所的な例外処理も現実世界に対応づけたルールで記述できるので，エンドユーザにわかりやすい．

4.2 従来方式との比較

上記のルールを従来のスクリプト言語を用いて記述した場合は，以下のようにしてメッセージセットを記述することになる．

```
public 書類受理 (CFP) {
  ...
  [ (作業継続 == true && 印刷の有無 == true)
      [ 印刷. 印刷 (CFP), this. 印刷終了通知 () ],
    (作業継続 == true && 印刷の有無 == false)
      [ CFP 配布. 配布 (CFP), メール送信. 送信 () ] ];
}
```

しかしながら，このような記述は直感的に理解することが難しく，ある程度のスキルを必要とするためエンドユーザ向きではない．

また，制御コンポーネントを用いてビジュアルに定義した場合を図 5 に示す．この方法では，見た目は分かりやすいが，１つの分岐を表現するために条件判定コンポーネントやイベントコンポーネント等を配置し，さらに各コンポーネント間の引数の受け渡しなども全て定義しなければならないため，ユーザへの負担が大きくなる．

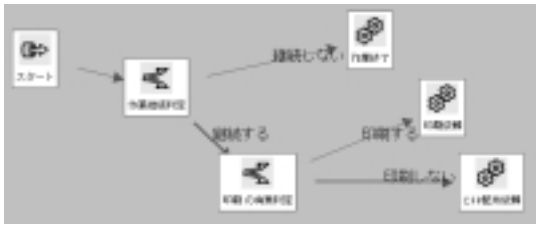

図 5 制御コンポーネントによる定義

ルールを用いた分岐記述の場合は，業務の各ステップにおいて業務知識をそのままルール化すればよいので，エンドユーザでも簡素でかつ柔軟なワークフローを自ら定義できる．また，ルールは追加，削除，変更などが比較的容易に行えるため保守にも有効である．

5. ユーザインタフェースの構築と検証

5.1 UI の自動生成

一般に，アプリケーション開発におけるユーザインタフェースの構築作業は開発工程全体の中で多くの時間を費やし，変更や修正も頻繁に発生する [4]．さらにエンドユーザ主導型開発である **M-base** においては，開発中に要求仕様やドメインモデルの変更が頻繁に起こることが予想されるため，それに影響されて UI の変更も頻繁に発生する．しかしこれらの変更が起こる

図 6 自動生成された UI

度に UI を修正していては，エンドユーザにとって多大な負担となる．そこで **M-base** では，UI を自動生成することによりエンドユーザの負担を軽減する．

M-base における従来の自動生成の方式では，出力項目に対してラベルが，入力項目に対してテキストフィールドが単純に配置されるだけであった．しかしこの程度の自動生成では，使い勝手が不十分であり，入力ミスも多く発生するため，実際のアプリケーションに用いるには多くの部分をカスタマイズする必要があった．そこで，出来るだけ使い勝手の良い UI を作成するために，次のような 3 段階方式を採ることにした．

(1) 入出力情報の定義
(2) UI の自動生成
(3) カスタマイズ

まず UI における入出力情報とそのタイプ（日付型，整数型，真理値型など）を指定する．次に UI ビルダはその情報をもとに，適切な部品を配置して UI を自動生成する．ここでは日付型であればカレンダーウィジェットを，真理値型であればチェックボックス等を配置することにより，使いやすい UI を自動で生成できるようにした．さらに必要であれば，生成された UI に対して部品の位置やサイズを変更したり，部品間の関連付けを行う．部品間の関連付けでは「テキストを入力しないとボタンが押せない」というような動作を UI に組み込むことが可能である．

図 6 は Call for Paper を作成する際に用いられる UI を自動生成したときの画面である．ここで自動生成された UI は，デバッグ作業にも用いることができる．すなわち **M-base** では，アプリケーション開発中は自動生成された UI を用いて検証作業を繰り返し，完成後は使い勝手向上のためにカスタマイズを行うことにより，モデルの変更に伴う UI の修正作業を軽減している．

5.2 UI 遷移図を用いた検証

一般にアプリケーション開発の初期段階において，ユーザインタフェースの詳細や画面遷移などを実装し，アプリケーションを利用するユーザに実際に使用してもらう方法が従来から行われてきた．これは，操作性の確認やユーザの漠然とした要求を早期に具体化することを目的としている．

さらに **M-base** では，アプリケーション開発者がエンドユーザであるため，曖昧な要求を自分で具体化し，その要求を自分でモデルに反映させなければならない．そこで，作成したモデルから UI 遷移図を表示することにより，エンドユーザが自ら要求の検証や洗練を行うことを可能にした．またシステム全体が完成していなくても，遷移図を生成することができるので，部分的な検証を行うことも可能である．

図 7 は，スケジュール決定から CFP 作成までの UI 遷移図である．この遷移図から必要項目の欠如や，作業手順の間違いなどを発見し修正することができる．

図 7 UI 遷移図による検証

5.3 UI 遷移図を用いた検証事例

例題において，生成された UI 遷移図をもとに，ドメインモデルを改良した結果について以下に示す．

(1) スケジュール決定後，すぐに CFP の作成を行うとは限らないので，作業を継続するかどうかを入力できるように UI を修正し，そのためのルールを追加した．
(2) 遷移図上で CFP 作成画面から CFP 確認画面へ移行していないという不具合から，CFP 作成コンポーネントのルールの記述ミスを発見し，修正した．
(3) 委員を登録した後，続けて複数の委員を登録できるようにするために UI を変更し，そのためのルールを追加した．

以上から，遷移図を表示することにより，ユーザの要求を洗練したり，モデリングの誤りを発見することが可能となり，UI 遷移図の有用性を確認できた．

6. シミュレーション検証

6.1 ３種類の図式表現を併用した検証

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．シミュレーションもユースケース単位で行う．業務フローの各メッセージをステップ実行することにより，エンドユーザによるシミュレーション実行を容易にした．また必要に応じて UI から実際に値を入力することができるため，UI とモデルの連係処理もチェックできる．

シミュレーションによる検証では，ドメインモデルをそのまま用いる方法（コラボレーション図），事象トレース図を用いる方法（シーケンス図），プロパティシートを用いる方法の３種類を併用して行う．

コラボレーション図（図 4 左）は，メソッドが起動されるたびにメッセージを表す矢印と実行履歴を表す数字を表示していく．ユーザが作成したモデル上で動作の確認が行えるため，直感的に理解しやすい．また，並列処理を表現するのにも優れている．

シーケンス図（図 4 右）は，オブジェクト間通信を時系列で表示するものである．ここではメソッド名だけを用いた動作確認には限界があるため，モデリング時に業務の用語を用いて詳細化した処理内容を表示することにより，視覚的に判りやすくした．

プロパティシート（図 8）は，実行中のオブジェクトプロパティの値を表示または変更するためのツールである．上記のコラボレーション図とシーケンス図が主として業務フローの確認に用いられるのに対して，プロパティシートは実際の値を確認しながら，より詳細な検証を行うために用いる．プロパティ値の確認とともに動的に値を変更することもできるため，シミュレーションの効率も向上する．このようにプロパティシートを用いた検証を行うことにより，記述したスクリプトが正しく実行されているか，UI から入力した値がモデルに正しく反映されているか，ルールが正しく選択されているかなどのより厳密な検証を行うことが可能になった．

このような３種類のシミュレーション方法の併用により，エンドユーザ自身が自ら作成したモデルの検証を行える．

6.2 シミュレーションによる検証事例

例題において，３種類の図式表現を併用したシミュレーション検証を行った結果について以下に示す．

(1) スケジュールを登録したが，プロパティシートに値が反映されていなかったため，スケジュール表コンポーネントを調べてみたところ，プロパティ名を誤って入力していたことが判明したので修正した．

図 8 プロパティシートを用いた検証

(2) スケジュール決定コンポーネントのプロパティ「作業継続」の値が true になったことを確認したが，作業が終了してしまったのでルールを調べたところ，記述ミスを発見したので修正した．

このように，プロパティシートを導入したことにより，アプリケーション内部の詳細な動きまで確認することが可能になった．

7. コンポーネントに関する考察

M-base では業務コンポーネントの組合せを基本としているので，コンポーネントの開発・利用形態が重要である．そこでコンポーネントの開発・支援方法を検討するために，**M-base** で扱うコンポーネントを図 9 のような４つに分類した．図の縦方向はコンポーネントを利用する際のユーザの定義量による分類であり，横方向はコンポーネントの適用範囲による分類である．以下に，これらの４つの部品群に関して **M-base** における開発・支援方法を考察する．

基本部品は，各分野に共通して用いることができる汎用的なコンポーネントである．例題では，印刷，メール送信，書類作成に相当する．このような部品は比較的数が限られており使用頻度が高いため，あらかじめ **M-base** で提供している．これらの部品はカスタマイズする必要がなくブラックボックスとして扱うことができるため，エンドユーザによる利用は容易である．

特定分野部品は，特定業務の分野においてのみ利用されるコンポーネントである．例題では，委員名簿管理，投稿論文管理に相当する．これらの部品は，多くのアプリケーションを構築していく過程で蓄積されていく．これらが充実してくると，エンドユーザによる構築が容易になる．

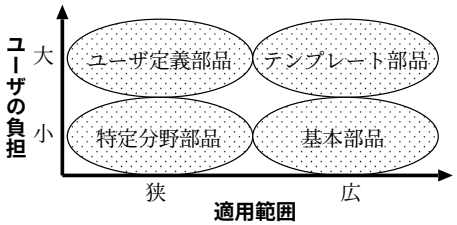

図 9 コンポーネントの利用形態による分類

ユーザ定義部品は，個々のアプリケーションに特化したコンポーネントである．例題では，スケジュール決定，CFP 作成，CFP 配布，委員選出，投稿論文受付に相当する．これらの部品は事前に用意することが難しく，現時点ではユーザ自身が新規に定義することになる．**M-base** では，このようなオブジェクトの作成を支援するために，動的モデルから必要となるオブジェクトのスケルトンを自動生成することで，エンドユーザによる構築を容易にしているが，システムエンジニアの支援を受けることもあり得る．

テンプレート部品は，基本部分はあらかじめ定義しておき，ユーザが利用時にカスタマイズするコンポーネントである．例題では，スケジュール表に相当する．これらの部品は **M-base** であらかじめ提供するのが望ましいが，当初は必要に応じてシステムエンジニアが開発したものを蓄積していくことになる．

以上の４つの部品群に加えて，これらの部品の組合せ技術も重要である．最近では，見かけ上の部品の粒度を大きくして使いやすくする技術として，アプリケーション・フレームワーク，デザインパターンなどが注目されている．**M-base** では，エンドユーザ主導の開発という観点から，これらの技術をベースにした業務モデルと業務コンポーネントの対応づけが重要である．一方，業務モデルを特定できる場合は，予め対応づけされたコンポーネント集合をアプリケーションフレームワークとして提供する方法を検討している [14]．

8. おわりに

本稿では，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．特にルールを用いたフロー分岐の記述，ユーザインタフェースの自動生成および遷移図を用いた要求の洗練，シミュレーションツールによる厳密なモデル検証について述べた．

本システムの実装には Java 言語を用い，スクリプト言語には Pnuts[15] 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 120 クラスで約 13,000 行，UI ビルダは 25 クラスで約 1,500 行，アプリケーション実行部は 40 クラスで約 4,300 行である．

謝辞 本稿をまとめるにあたり，有益なアドバイスをいただいた査読者の方々に感謝致します．

参 考 文 献

1) 青山幹雄, 中所武司, 向山博: コンポーネントウェア, 共立出版 (1998).
2) Jacobson, I., et al.: *Object-Oriented Software Engineering*, Addison-Wesley (1992).
3) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-Oriented Modeling and Design*, Prentice-Hall (1991).
4) Daniel Takach, Walter Fang, Andrew So: *Visual Modeling Technique*, Addison-Wesley (1996).
5) Unified Modeling Language Resource Center: http://www.rational.com/uml/index.jtmpl
6) TriReme Component Based Development – Catalysis: http://www.trireme.com/trireme/catalysis/home.htm
7) 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).
8) 中所武司, 小西裕治, 松本光由: 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, 情報処理学会論文誌, Vol.40, No.5, pp.2387-2396 (1999).
9) 岩田智彰, 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** におけるコンポーネントベース開発技法, ソフトウェア工学研究会, No.98-SE-120-18 (1998).
10) 石榑久嗣, 中所武司: メッセージフローに基づくアプリケーション開発技法, 情報処理学会第 60 回大会講演論文集（1）5ZB-01, pp.317-318 (2000).
11) 紺田直幸, 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** における開発・検証技法, ソフトウェア工学研究会, No.4-SE-125-3 (2000).
12) 情報処理学会要求工学 WG 共通問題: 国際会議のプログラム委員長の業務, http://www.selab.cs.ritsumei.ac.jp/~ohnishi/RE/problem.html
13) 戸田保一, 飯島淳一, 速水治夫, 池内正博: ワークフロー ビジネスプロセスの変革に向けて, 日科技連 (1998).
14) 藤原克哉, 中所武司: 窓口業務を例題としたエンドユーザ向き分散アプリケーションフレームワーク wwHww の開発と適用評価, 情報処理学会論文誌, Vol.41, No.4, pp.1202-1211 (2000).
15) Pnuts Quick Start: http://javacenter.sun.co.jp/pnuts/index.html