3．4 M-base：「ドメインモデル≡計算モデル」を志向した開発技法 (中村作)

3．4．1 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。本技法では、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。本節では、本技法の目的と対象、モデリングプロセスの概要、モデリング＆シミュレーション、スクリプト言語の特徴、特定分野向きのコンポーネントの抽出方法などについて述べる。

3．4．2 目的と対象

(1) エンドユーザ

一般に、エンドユーザの範囲は広いが、本技法では、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) 対象ソフトウエア

本技法では、「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。

(3) 開発・保守形態

本技法では、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。

3．4．3　モデリングプロセスの概要

(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。ここでは、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。

(a) 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。

(b) 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。

(2) ドメインモデルの構築手順

本技法では、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[4]を行う。概要を図3.4-1に示す。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。

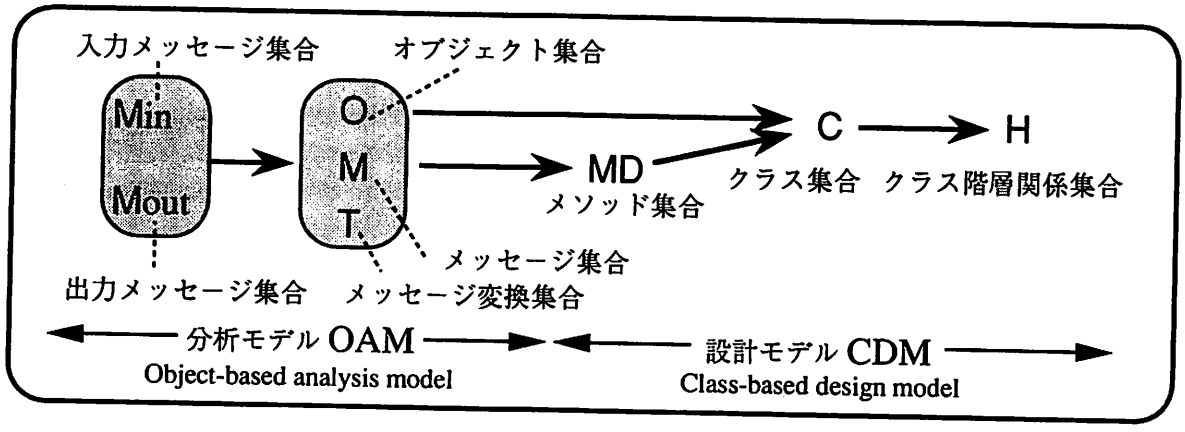

図3.4-1 M-Baseのモデリングプロセス

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。

3．4．4　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3.4-2に示す。開発環境は主に次のような構成要素からなる。

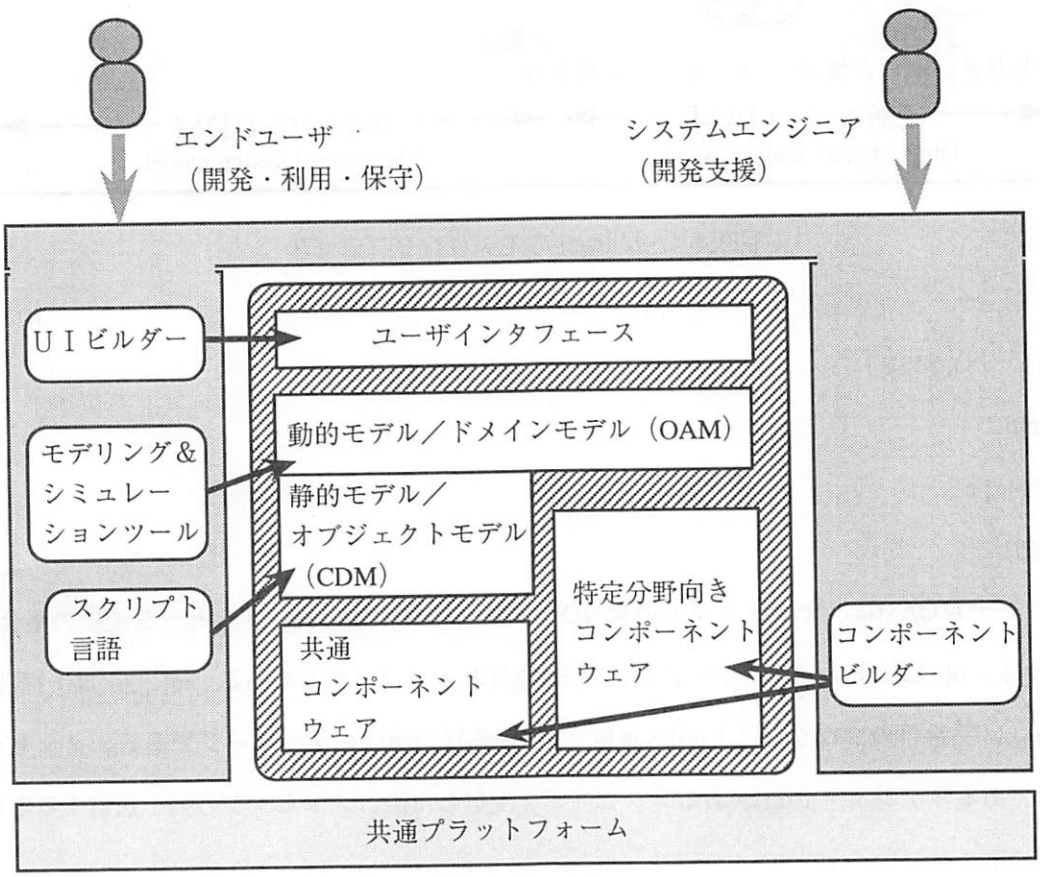

図3.4-2　M-baseの開発環境（外側）とアプリケーション・アーキテクチャ（内側）

・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。

3．4．5 モデリング＆シミュレーション

(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。

(a) 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。

(b) 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。

(c) 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。

本技法では、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。

(2) 基本機能

モデリング＆シミュレーションツールは、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。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3.4-3が画面の例である。この図は、左上の事務局オブジェクトが外部から会議開催準備のメッセージを受け取ると、右側の会議室オブジェクトやOHPオブジェクトに予約のメッセージを送り、左下の安部さん、馬場さん、千葉さんのオブジェクトに会議開催案内のメッセージを送るというモデルを作成しているところである。

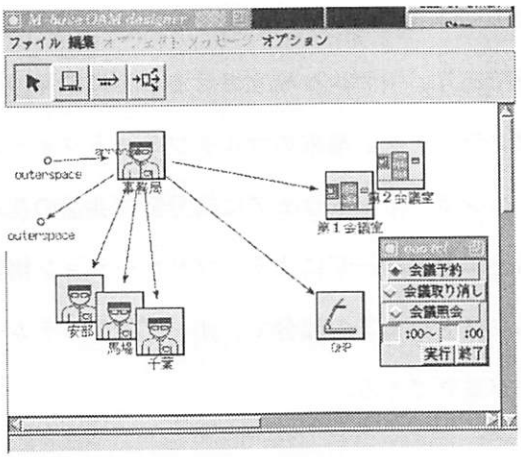

図3.4-3 M-baseによるモデリングの例

主な基本機能を以下に示す。

(a) オブジェクトの生成と表示

(b) メッセージの生成と表示

(c) メソッド定義

(d) メッセージ属性参照、変更

(e) メッセージ送信

(3) システム開発の手順

シミュレータでは、メッセージ変換の設定によって、SendMessageを含んだメソッドを自動的に作り出す。システム開発の手順は次のようになる。

(a) オブジェクトとメッセージを定義する。

(b) オブジェクトがあるメッセージを受け取り、その処理の中で幾つかのメッセージを送信するしくみをメッセージ変換として定義すると、SendMessageを含んだメソッドが自動生成される。

(c) 必要に応じてメソッド中に命令を追加する。

たとえば、図3.4-4に示すように、OfficeオブジェクトにArrangeメッセージを入力し、RoomオブジェクトにReserveメッセージ、AbeオブジェクトにAnnounceメッセージを出力するモデルを作成した場合、次のようなメソッド定義が自動生成される。

```
proc Office_Arrange { } {
    SendMessage Room Reserve
```

```
    SendMessage Abe Announce
  }
```

このメソッド定義に対し、会議室が予約できた場合だけ会議開催案内を出すようにユーザが変更を加えても良い。一例を以下に示す。

```
proc Office_Arrange {meeting day hour} {
  SendMessage Room Reserve $meeting $day $hour
    if {$result == "SUCCESS"}
      then {SendMessage Abe Announce $meeting $day $hour}
}
```

シミュレーションの実行例を図3.4-4と図3.4-5に示す。図3.4-4は、外部からの会議開催準備の指示メッセージを入力するために、吹きだしの形式でArrangeメソッドのパラメータ要求表示をして、会議名と日時を設定した画面である。この後、メッセージ送信をメニューで指示すると、図3.4-5の画面になる。この画面は、事務局オブジェクトが会議室予約管理オブジェクトに会議室予約のためのReserveメッセージを送信するところである。

開発済みのプロトプログラムは、実装はTcl/Tkを用いて行ない、作成したモデルから、Tcl/Tkファイルを出力するようにしている。現在、Java言語の版を開発中であり、スクリプト言語も次に述べるように、エンドユーザ向きのものを開発中である。

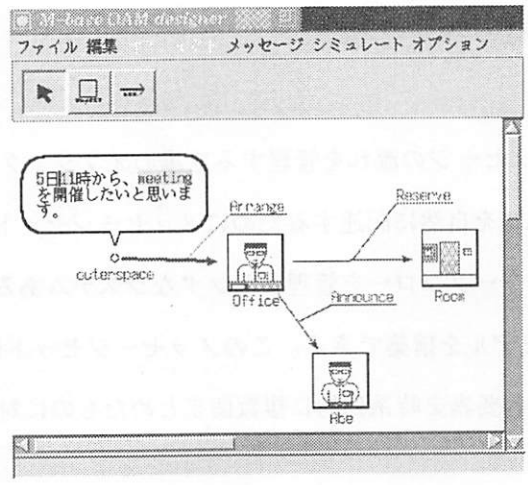

図3.4-4　M-baseによるシミュレーションの例（メッセージ送信前）

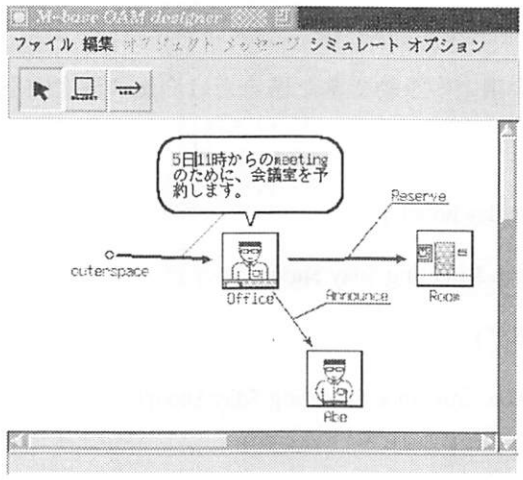

図3.4-5 M-baseによるシミュレーションの例（メッセージ送信後）

3．4．6 スクリプト言語

(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。

(a) 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。

(b) 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。

(2) 分散協調型モデルの表現方法

(a) メッセージセット

Hoop のモデルでは、本来メッセージの流れを管理する立場のオブジェクト（業務担当者）がオブジェクト間に伝わるメッセージの流れを自然に記述するためにメッセージセットを導入した。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。

(b) 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。

M ::= M' ¦¦ [cond] M'

M' ::= Ms ¦¦ Mp ¦¦ X

Ms ::= { M, M, ..., M }

X ::= (obj, msg, arg1, arg2, ... , argn) (n≧0)

objはオブジェクト、msgはメッセージ、condは条件、argn は引数、$\alpha \| \beta$ は α または β を意味する。

(c) 直列のメッセージセット

例えば、「ある書類を提出する時に、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」という場合を考える。このような逐次的に複数の処理が実行されるときは次のような直列のメッセージセットを記述する。

{ M1, M2, ... , Mn } (n≧1)

簡単な例として、{ (o1, m1), (o2, m2), (o3, m3) }というメッセージセットがo1送られたとする。オブジェクトo1はメッセージm1を実行後、オブジェクトo2に対して、{ (o2, m2), (o3, m3) }を送る。o2は同様にm2を実行後、o3に{ (o3, m3) }を送る。

(d) 並列のメッセージセット

次に、「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」という場合を考える。このように複数の処理が並行して実行されるときは次のような並列のメッセージセットを記述する。

{ M1 ¦ M2 ¦ ... ¦ Mn } (n≧1)

簡単な例として、{ (o1, m1) ¦ (o2, m2) ¦ (o3, m3) }という並列メッセージの場合は、オブジェクトo1、o2、o3にそれぞれメッセージm1、m2、m3が送られる。

(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。その概要は図3.4-6に示す。

(a) 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用いてモデルを記述する。

(b) 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。

(c) 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。

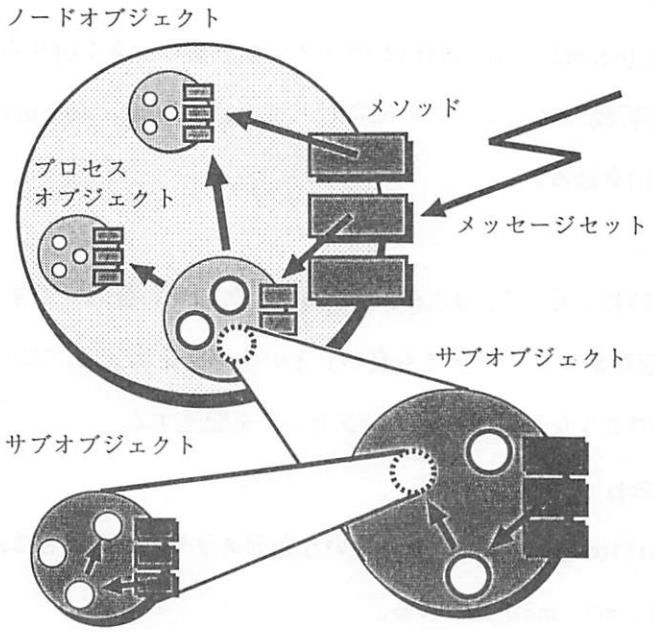

図3.4-6　3種類のオブジェクト

3．4．7　コンポーネントウェア抽出実験

(1) エンドユーザ主導の開発

エンドユーザ主導の開発における課題として、第1にコンピュータの専門的な知識の不足がある。そのため、エンドユーザにはコンピュータの専門的な知識を必要とする部分を見せないようにするシステムの支援が必要である。第2にはソフトウェア部品の粒度の問題がある。以下に示すような、アプリケーションを構成するソフトウェア部品の粒度と利用者との関係に注意しなければならない。

(a) 部品の粒度が小→利用者の負担増加

(b) 部品の粒度が大→利用者の自由度減少

(2) M-baseでのコンポーネント

現在、オブジェクト指向の世界では「ソフトウェア部品の粒度の問題」を解決するものとして、コンポー

ネント、フレームワーク、パターンといった研究が注目をあびており、各方面での研究が盛んである。コンポーネントとしては、基本コンポーネントと特定業務向けコンポーネントの２種類を考える。基本コンポーネントとは、情報システムを設計するうえで汎用的に利用できる部品である。例えば、GUI部品や基本データ型等がこれにあたる。基本コンポーネントだけでアプリケーションを開発する際に問題となるのが、業務モデルのオブジェクトに対応した適切なコンポーネントを選択することが難しいことである。そこで、特定業務領域に限定したコンポーネントを用意することが、これらの問題を解決するものとして有効であると考えている。

(3) 部品抽出の事例

この特定業務向けコンポーネントの抽出は、トップダウン的アプローチでは難しいので、業務の事例からボトムアップに行なう方法を支援する必要がある。部品抽出の事例として、次のような簡単な在庫管理を考える。利用者は、自分が食べたものを会計係にその都度申告する。会計係は、利用者からの申告を記録し、代金を集計し、月ごとに請求する。また、在庫が切れたり、利用者から欲しい商品のリクエストがあれば新しい商品を、買い出し係に発注する。買い出し係は、会計係から、購入依頼があったら買い出しに行く。また、会計係が管理しなければならないものとして、販売履歴表と商品管理表がある。

この業務内容のモデリングから、在庫管理システムに必要なコンポーネントとして次のようなものを抽出した。左側がモデルを構成するインスタンスオブジェクトであり、矢印の右側が抽出された部品である。

(a) 外部インタフェース→現実世界と計算機との橋渡しをする部品

(b) 販売履歴管理→名簿部品

(c) 在庫管理→商品管理表部品

(d) 会計係→受け付け部品

(4) 部品利用の事例

在庫管理システムで抽出したコンポーネントを会議開催事務処理システムに利用する場合を検討した。会議開催事務処理業務は、会議開催依頼を事務局が受け付け、必要な会議室や、OHP等の備品の予約管理を行ない、参加者に会議開催の通知をする、といったものである。　このモデリングを行い、先ほどの在庫管理で抽出されたコンポーネントの利用を検討した。左側がモデルを構成するインスタンスオブジェクトであり、矢印の右側が再利用される部品である。

(a) 外部インタフェース←現実世界と計算機との橋渡しをする部品

(b) 個人名簿管理←名簿部品

(c) 事務局←受け付け部品

会議室予約管理表と、備品予約管理表については利用できそうな部品が見当たらないので、新しく部品

を抽出する。これらは、管理する対象を時間で管理する部品であるので、スケジュール部品とする。

3．4．8 おわりに

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、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として、業務モデルと計算モデルの一致により分析、設計、プログラミングの一体化を実現する開発方法論を提案した。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支援するアプリケーション開発環境M-baseおよびそのモデリング＆シミュレーションツール、スクリプト言語などのツールについて述べた。今後は、共通プラットフォーム上にこれらのツールを統合すると共に、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、Vol.37, No.1, pp.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'96シンポジウム、pp.87-94, 1996.

3) 中所、小西、浜、吉岡：「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, 1996.

4) T. Chusho, Y. Konishi and M. Yoshioka : M-base : An Application Development Environment for End-users Computing based on Message Flow, APSEC'96, 1996.

5) E. Gamma, R. Helm, R. Johnson and J. Vlissides : Design Patterns, Addison-Wesley, 1995.

6) D. E. Monarchi and G. I. Puhr : A research typology for object-oriented analysis and design, Comm. ACM, vol.35, no.9, pp.35-47, 1992.