ソフトウェア工学 109－2
（1996．5．17）

「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

中所武司　小西裕治　濱　巨人　吉岡大生†

明治大学 理工学部 情報科学科
chusho@cs.meiji.ac.jp

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，

・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、そのためのモデリング&シミュレーションツールとして，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．スクリプト言語の設計においては，ネットワーク上のノード割り当て方法と並行処理の可否によってオブジェクトを３種類に分類すると共に，大規模化への対応としてオブジェクトのネスト構造を導入した．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することにより，全体のメッセージフローの管理を容易にした。コンポーネントウェアの活用のために特定分野向きのコンポーネントの抽出実験を行なった。

Application Software Development Environment, M-base, based on the Concept of "a Domain Model ≡ a Computation Model"

Takeshi CHUSHO Yuji KONISHI Naoto HAMA Masao YOSHIOKA
Meiji University, Department of Computer Science

One solution for application development by end-users, is a concept of "a domain model ≡ a computation model." A dynamic model on interactive behavior among objects is focused on prior to an object model. M-base supports this process by a modeling & simulation tool and a script language representing message flows with message sets.

†96年4月よりNEC情報システムズ勤務

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**[4] 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章ではモデリング&シミュレーション、５章ではスクリプト言語の特徴、６章では特定分野向きのコンポーネントの抽出方法について述べる．

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．

2.4 M-base の研究項目

M-base では、エンドユーザ自身の手による、比較的小規模なアプリケーションの開発を

支援する統合的な環境を実現するために、以下の分野の研究を行なっている。

- モデリング&シミュレーションツール [7]
- スクリプト言語 [2]
- コンポーネントウェア [6]
- ユーザインタフェースの設計

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 [4] 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．

$$\mathrm{OAM} = \{ \mathrm{O}, \mathrm{M}, \mathrm{T} \}$$
$$\mathrm{O} = \mathrm{o[i]}$$
$$\mathrm{M} = \mathrm{m[i,j,n]}$$
$$\mathrm{T} = \mathrm{t[r]}$$

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．

$$\mathrm{CDM} = \{\mathrm{MD}, \mathrm{C}, \mathrm{H}\}$$

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．

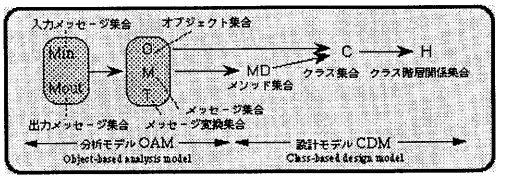

図 1: M-base のモデリングプロセス

4 モデリング&シミュレーション

4.1 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．

4.2 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2が画面の例である。

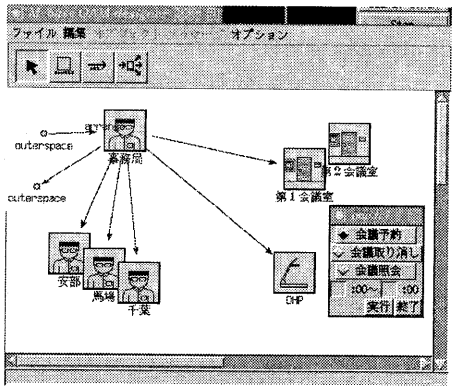

図 2: 開発中のモデリングツール

主な基本機能を以下に示す。

- オブジェクトの生成と表示
- 外部オブジェクトの生成と表示
- メッセージの生成と表示
- メソッド定義
- メッセージ属性参照、変更
- メッセージ送信

4.3 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。

1. オブジェクトとメッセージを定義する。
2. 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。
3. 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と図 4に示す。図3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4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。

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。

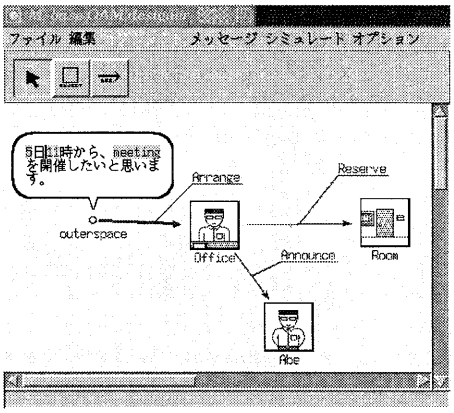

図 3: 実行例（メッセージ送信前）

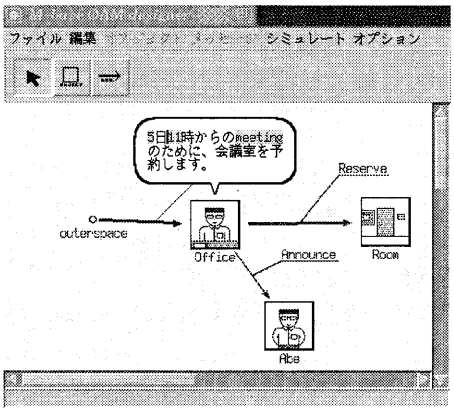

図 4: 実行例（メッセージ送信後）

5 スクリプト言語 Hoop

5.1 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。

5.2 分散協調型モデルの表現方法

5.2.1 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。

5.2.2 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 [3] に詳しい。

$$
\begin{aligned}
M &::= M' \| [cond] M' \\
M' &::= M_s \| M_p \| X \\
M_s &::= \{M, M, \ldots, M\} \\
M_p &::= \{M | M | \ldots | M\} \\
X &::= (obj, msg, arg_1, arg_2, \ldots, arg_n) \ (n \geq 0)
\end{aligned}
$$

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

5.2.3 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。

5.2.4 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。

5.3 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(図5)。

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。

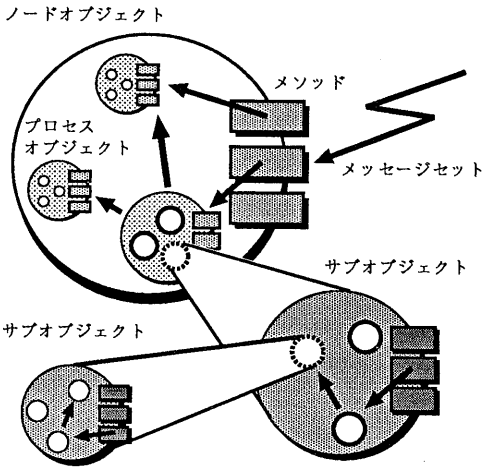

図 5: 3種類のオブジェクト

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 コンポーネントウェア抽出実験

6.1 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。

- 部品の粒度が小→利用者の負担増加
- 部品の粒度が大→利用者の自由度減少

6.2 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た。

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考える。

6.3 部品抽出の事例

部品抽出の事例として、報告者らの大学の研究室で実際に行なっている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。以上のことをまとめた業務モデル図を図6にしめす。

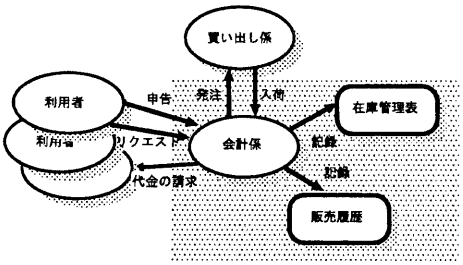

図 6: 在庫管理業務モデル図

このモデルのうち、会計係の業務(図6の影の部分)をコンピュータ化する例を考える。業務内容と業務モデル図から、在庫管理システムには以下のようなコンポーネントが必要だと考えられる。

現実世界と計算機との橋渡しをする部品
: 利用者や、買い出し係などコンピュータ化するシステムの外側のものは、コンピュータ化したシステムとメッセージのやり取りを行う。実際には、電子メールやインターネットなどを利用し、コンピュータ化したシステムと現実世界との橋渡しの役割をするコンポーネントである。コンポーネントの名前と送受信するメッセージをカスタマイズすることにより、変更できる。

名簿部品
: 販売履歴表は、誰が(人)、何を(商品)、どれだけ(数量)買ったかを記録する表である。対象となる業務領域での人間の名簿として提供される部品であり、各人についての項目を追加できる。

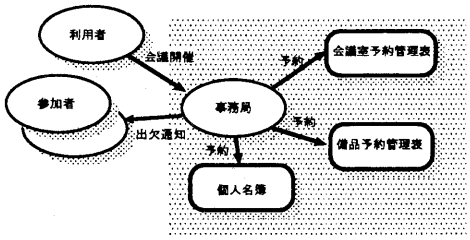

図 7: 会議室予約管理業務モデル図

商品管理表部品
: 商品管理表は、商品名、仕入れ価格、販売価格、在庫の数量を記録する表である。商品名から、仕入れ価格、販売価格、在庫の数量が検索できるような仕組みが必要である。

受け付け部品
: 会計係は販売履歴表と、在庫管理表を参照して、メッセージにしたがってそれぞれの表を更新する。仕事の要求を受け付け、適切な処理をする。

6.4 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室予約管理システムに利用する場合を検討した。会議室予約管理業務は、会議開催通知を事務室が受け付け、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。図7に会議室予約管理業務モデル図を示す。

このモデルのうち、事務室の業務 (図7の影の部分) をコンピュータ化する場合を考えて、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する。利用者、参加者が「現実世界と計算機との橋渡しをする部品」、個人名簿が「名簿部品」、事務室が「受け付け部品」にあたる。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。

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** の開発技法として、モデリング&シミュレーションツール、スクリプト言語、コンポーネントウェア機能について述べた。今後は，これらの技術を統合すると共に、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: オブジェクト指向分析・設計言語 Hoop における分散協調型モデルの表現方法, 日本ソフトウェア科学会 第 12 回大会論文集 pp. 197-200, 1995.

[3] 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（発表予定）1996.

[4] 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-104-4，1995.

[5] 中所 武司：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，平成 7 年度 EAGL 基礎・育成助成研究成果報告書，93-104, 1996.

[6] 浜 巨人：アプリケーションソフトウェア開発環境 M-base ー特定分野向けアプリケーションのためのコンポーネント部品の抽出と利用ー, 明治大学理工学部情報科学科、中所研究室、１９９５年度卒業論文, 1996.

[7] 吉岡 大生：アプリケーションソフトウェア開発環境 M-base ーモデリング&シミュレーションツールー, 同上.