データベースシステム 89−12
（１９９２．７．２３）

オブジェクト指向ネットワーク環境データベースにおける自律オブジェクトについて

福田 健一* 村田 美恵* 明田 行史* 布川 博士** 増永 良文***

*AIC **東北大学電気通信研究所 ***図書館情報大学

コンピュータネットワークの普及にともない，ネットワークの大規模化・複雑化が進展している．そのためユーザや管理者がネットワーク資源を把握し活用できるように，ネットワークだけではなく，ソフトウェアや管理情報，そしてヒューマンファクタをも含めたネットワーク環境に関する情報システムの構築が重要となってきている．我々は，オブジェクト指向データベースを用いて，ネットワーク環境に関する情報を適切に提供できるシステムを構築中であるが，本稿ではネットワーク環境に関するデータの中で，特に動的に変化するデータをネットワーク環境データベースに取り込むモデルについて述べる．実現方式の検討と試作例の結果より，スキーマ定義や応用プログラムとのインタフェース規定等に本モデルが有効であることを示す．

Autonomous Objects in Object-Oriented Network Environment Database

Ken'ichi Fukuda* Mie Murata* Yukifumi Akeda*
Hiroshi Nunokawa** Yoshifumi Masunaga***

*Advanced Intelligent Comm. Sys. Lab.
**Research Institute of Electric Comm., Tohoku Univ.
***Univ. of Library and Information Science

6-3-3, Minamiyoshinari, Aoba-ku, Sendai 989-32, Japan

As the computer networks proliferate, they are getting larger and more complex. It is important to provide the information system about the network environment for users and administrators who want to understand and efficiently use all network resources. The computer network environment should include not only the network hardware but also the software, the administration information and the human factor. We are developing the network environment database using the Object-Oriented Database technology. In this paper, we describe the model which enables the network environment database to incorporate dynamic data. Then we represent the efficiency of this model through the implementation of the prototype system.

1. はじめに

ワークステーションをはじめとするコンピュータネットワークが拡大・普及し，ネットワークの複雑化，多様化の進展が著しい．ネットワークの拡大により，その管理者でさえもネットワークの各種の状況を十分に把握できないのが現状である．また，ネットワークの普及にともない，必ずしもネットワークに関する知識を有している人々だけが管理者になるわけではなくなってきており，管理者層もまた多様化している．同様に，ユーザにおいても，ネットワークに関する知識なしではコンピュータを利用することが困難になってきており，ユーザの層もまた多様化している．

このような状況のもとでは，ネットワークを単なるコンピュータ結合網と考えるのではなく，ソフトウェアや管理情報，そしてヒューマンファクタをも含むネットワーク環境として捉える必要がある．さらに，多様化した管理者やユーザに対してネットワークに関連する種々の情報を適切に提供するシステムが必要となってきている．そのため，このシステムはユーザインタフェースをも包含するシステムでなくてはならない．以上の問題を解決するために我々はネットワーク環境をオブジェクト指向データベース(OODB)によってモデル化し，それを核として，ユーザインタフェースのレイヤまでも考えたシステムを構築している[1,2,3,4,5,6].

本システムのアプリケーションには，(1) ネットワーク管理・設計支援，(2) ネットワーク環境の状況説明，(3) ネットワーク利用のシミュレーションを想定している．そのため，ネットワーク環境のデータとして，時間変化のほとんどない静的データのみならず，ネットワーク使用中に刻々変化していく動的データをも取り扱える必要がある．

本稿では，OODBの枠組みの中で，動的データを静的データと同等に取り扱えるモデルを示す．また，スキーマ定義と応用プログラムとのインタフェースの試作例についても述べる．

2. ネットワーク環境データベースと実世界

我々は，1章で述べた目的をもって，ネットワークの接続関係や，ワークステーションなどのハードウェア構成，また，ソフトウェアやネットワークの利用者をモデル化し，クラス構造(Class Structure)などを作成している．このモデル化にはCoad / Yourdon 法を利用した[7]．モデル化の結果に基づいて，スキーマを定義し，ネットワーク環境データベースのプロトタイプシステム LANDB を作成している．

ここでのモデル化の対象は，各機器の構成・設定・接続関係など時間変化の少ないデータ，すなわち静的データであった．1ヵ月，1年のオーダでしか変化しないこれらのデータは，小規模のネットワークにおいては人手による登録・更新が可能である．

これに対して，実世界には比較的短い時間間隔で変化する動的データが存在する．例えば，あるワークステーションにログイン中のユーザやディスクの空き容量に関するデータである．また，ワークステーションの移動やルータの追加など，ネットワークやシステムの構成設定が頻繁に変更されることもある．これらの動的データをデータベース上で人手により管理することは不可能である．

そこで，図1に示すように，実世界をネットワーク環境データベースに反映させる方式が問題となる．実世界の変化は，オブジェクトの追加・削除，またはオブジェクトの内部状態の更新に対応づけられるが，本稿では特にオブジェクトの内部状態の更新について考察する．

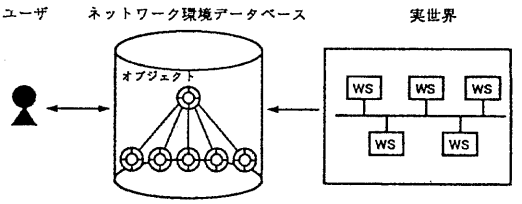


図1　ネットワーク環境データベースと実世界

3. 実世界データの自律的な取り込み

ネットワーク環境データベースにおけるオブジェクトの自律性とは「実世界の変化に従って，自分自身で内部状態を更新すること」を指す（図2）．

この自律性が実現できれば，(1) ネットワーク管理ソフトウェアなどの応用プログラム（データベースのユーザ）がリアルタイムデータを取り出せるようになる．(2) 人手を介さないでデータベースへデータを取り込めるようになる．(3)ネットワーク管理システムが 動的データの履歴もとることができるように

なる．(4)動的データを静的データと一緒に処理するための手段を提供できるようになるなどの利点が生まれる．

また，スキーマ設計においても，動的データを静的データと同等に扱えるようになるため，動的・静的の区別を意識しない自然なモデル化が行えるようになる．

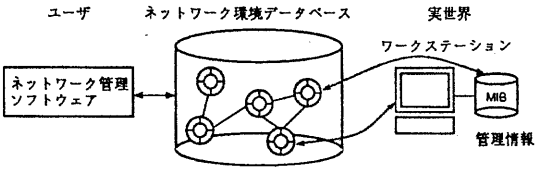


図2　実世界データの自律的な取り込み

4. 自律オブジェクト

4.1 自律性のモデル

自律オブジェクトのモデルを図3に示す．OODB中のオブジェクトは，内部状態を表す変数とユーザからの（あるいは他のオブジェクトからの）メッセージパッシングを受け付けるためのメソッドをもつ．更に，自律オブジェクトは，実世界の動的データに対応している変数を実世界からのメッセージパッシングにより更新するためのメソッドをもつ．このメソッドはユーザには非公開とする．

例えば，内部状態として2つの変数X, Yをもつ自律オブジェクトを仮定する．accessX, accessYは，それぞれXとYの値を返すメソッドである．応用プログラムや，データベース中の他のオブジェクトからのメッセージパシングにはaccessX, accessYを使う．一方，実世界において変化があると，その変化は，ユーザからは見えないところでrealXを通してXの値へ反映される．

つまり，自律オブジェクトの内部状態は，ユーザからのメッセージパッシングと実世界の変化により更新される．したがって，自律オブジェクトは，ユーザから同じメッセージを受けても時に応じて違う応答を返すことになる．

本モデルは，応用プログラムから見ると，OODB中のオブジェクトが自律的に実世界のデータを収集しているように見える．内部状態が変化しても，メッセージに対する反応が変わるだけでOODBとしてはなんら不都合はない．

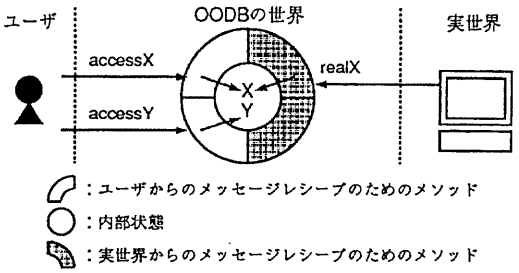


図3　自律オブジェクトのモデル

4.2 アクティブデータベースとオブジェクト

自律的な動作をするデータベースには，アクティブデータベースがある[7,8,9]．アクティブデータベースとは，データベース内部の状態変化を監視する機構をもち，状態変化をトリガとする処理を記述できるようにしたデータベースである．応用プログラムから見て，データベースが能動的に処理を実行する点は，本論文の自律オブジェクトと共通しているが，実世界の変化を取り扱っていない点で異なっている．

また，プログラミング言語や計算モデルの世界では，自律的な動作をするオブジェクトとしてアクティブオブジェクトなどがある[10,11]．アクティブオブジェクトは，他のオブジェクトからのメッセージパッシングとは独立して処理を実行するオブジェクトであると一般に定義できる．例えば，他からの処理を受け付けながら，自己の内部状態を常に監視していて，異常な状態になると正常化のための処理を並行に実行するようなオブジェクトである．これは，オブジェクトのメソッドに2つの種類がある点で，本論文の自律オブジェクトと共通しているが，実世界を写し込んでいるのではない点が異なる．

5. 自律性の実現方式

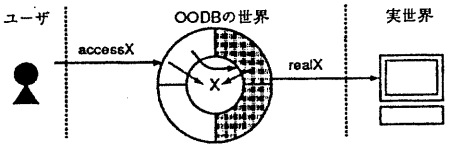
図4(a)は個々のオブジェクト毎に自律性をインプリメントする方法である．実世界のデータを取得し，その値を返すようなメソッドを，動的データに対応する内部変数毎に用意する．その内部変数にアクセスするようなメッセージを受けたときには，内部変数を参照する代わりに，実世界データの取得メソッドを実行した結果を返すようにする．また，そのメソッドだけでなく，他のメソッドがその変数を参照している場合も，内部変数ではなく実世界データ取得メソッドを使うようにする．

図4(a)に示すようにユーザがaccessX により内部変数X にアクセスする場合は，まず実世界のデータを取り込むrealX を実行し，その値をX に設定した後で，X の値を返す．他のメソッドが変数X を参照する場合もrealX を実行するようにする．ユーザからはaccessX を用いて変数X にアクセスしているようにしか見えない．

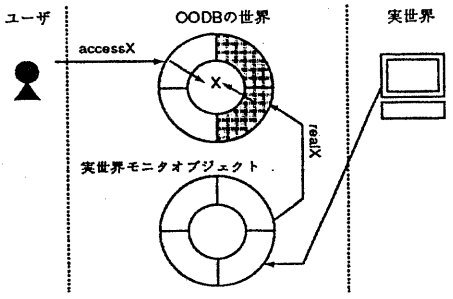
この方法では，ユーザが必要とするときだけ内部状態を更新するため効率はよいが，障害監視システムなどの実世界の変化をトリガとする処理が取り扱えない．このような場合には別の方法を用意する必要がある．

そこで，図4(b) に示すように実世界側に変化をデータベースへ伝える機構があるとする．データベース内でそのメッセージを受け付けるオブジェクトを用意する．これは実世界監視のためのオブジェクトである．メッセージを受け取ったこのオブジェクトは変更する必要のあるオブジェクトのrealX を用いて内部変数X の値を変更する．更に，必要ならば内部変数X の値が変わったことを通知するためのメッセージを送出する．

この方式では，応用プログラムからあまりアクセスされないオブジェクトも刻々と内部状態が更新されるため，実世界の監視を必要としない場合には効率的でない．



(a) 実世界とメッセージをやり取りするメソッドがある場合



(b) 実世界をモニタするオブジェクトがある場合

図4　自律性の実現構造

6. システムの試作

我々の作成しているプロトタイプシステムLANDB上で自律オブジェクトを試作した結果について二例述べる．どちらも実世界のワークステーションに対応するWorkstation オブジェクトに自律性を持たせたものである．ひとつは，ネットワークを管理するためのデータである管理情報をリアルタイムに取り込むものであり，もうひとつは，ログイン中のユーザの変化をモニタし，ディスプレイ表示を更新するものである．

試作にはONTOS[12], XView[13], C++ を用いている．

6.1 MIB の取り込み

MIB (Management Information Base) の仕様は多くのInternet ホストの監視，制御のための変数（管理情報）を定義するためのものである．管理情報はMIBという仮想的な情報集積所を通してアクセスされる[14]．尚，管理情報はオブジェクトとも呼ばれるが，OODB のオブジェクトと区別するために，ここではその呼称は使わない．

データベースのユーザに，実世界のデータである管理情報へアクセスする手段を提供する（図5）．ユーザは，accessMIB("管理情報識別子") というメソッドを介して，管理情報にアクセスする．

実世界から管理情報を取得するのに，SNMP(Simple Network Management Protocol)[16] を用いる．すなわち，realMIB の実行時にはWorkstationクラスのオブジェクトがSNMPマネジャになり，実世界のワークステーション上のSNMP エージェントと通信することになる．

Workstation のクラス定義をCoad / Yourdon法を用いて図6 に示す．MIB はWorkstation クラスの属性として現われ，静的なデータ(Name, CPUなど）と区別されることなしに列挙される．動的データであるMIB がクラス定義（スキーマ）へ取り込まれている．ここで，取り込むとはMIB へのアクセス手段を応用プログラムが知らなくてよいことを指す．すなわち，応用プログラムレベルではSNMP のマネジャ・エージェントモデルに関する知識が全く必要ない．

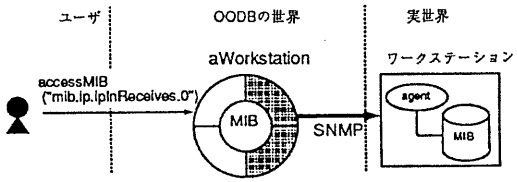


図5　MIB 情報のとりこみ

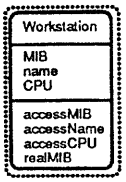


図6　Workstation クラス

6.2 GUI とのやり取り

ワークステーションの動的データを表示するポップアップウィンドウを作成する．動的データの例として，そのワークステーションに現在ログインしているユーザの名前一覧を取り扱う．

6.2.1 モデル

図7に示すようにWorkstation クラスの属性として，現在ログインしているユーザの名前一覧 currentUsers を定義する．実世界に変化があったときは，その属性値を表示している応用プログラムへ通知されるようにする．図中，PopupForWS がWorkstation の属性値を表示するポップアップウィンドウを生成するクラスであり，応用プログラムで定義されるものである．Refresh は Workstation に対してcurrentUsers の値を更新するよう指示を出すためのクラスであり，Workstation とともにOODB管理システムに管理されている．

図8(a) に，Workstationオブジェクトをデータベースから読みだして，ポップアップウィンドウに表示する時の処理の流れを示す．

①main() がPopupForWSのインスタンス(aPopupForWS) を生成する．

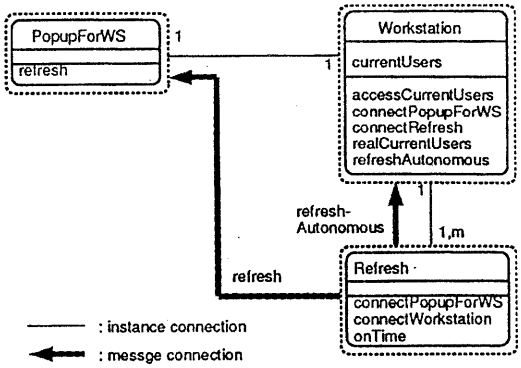
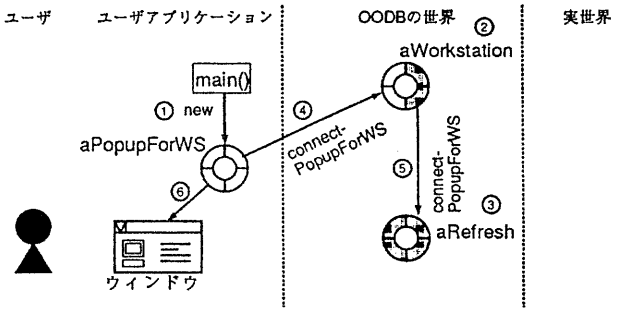
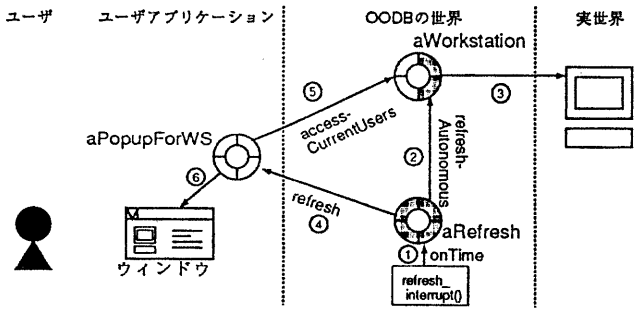


図7　GUI とのやりとり



(a) aPopupForWS 生成時



(b) refresh_interrupt() 起動時

図8　GUI とのやりとりの流れ

②aPopupForWSはWorkstation クラスのインスタンス(aWorkstation) をデータベースから読みだす．

③aWorkstation はRefresh クラスのインスタンス(aRefresh) をデータベースから読みだす．

aRefreshは時間の経過を通知する関数refresh_interrupt()を，ある時間間隔をおいて呼び出されるように登録しておく．

④aPopupForWS はaWorkstation へconnectPopupForWS メッセージを送る．

⑤aWorkstation はaRefresh へconnectPopupForWS メッセージを転送する．

⑥aPopupForWSはaWorkstation の属性currentUsersにアクセスし，ポップアップウィンドウに表示する．

図8(a)の後，実世界をモニタしているときは，図8(b)のような処理の流れになる．

①refresh_interrupt() が呼び出される．aRefresh へonTime メッセージが届く．

②aWorkstation へrefreshAutonomous メッセージを送る．

③aWorkstation はrealCurrentUsersを実行して 現在ログイン中のユーザを調べる．

④aRefresh はaPopupForWS へrefresh メッセージを送る．

⑤aPopupForWSはaWorkstation へaccessCurrentUsers メッセージを送り，currentUsers の変化に従ってポップアップウィンドウの表示を更新する．

ここで，応用プログラム，すなわちaPopupForWSは，aWorkstation が常に実世界の現在のcurrentUsersの値を保持していることと，変化があれば通知してくれることのみを知っていればよく，aRefresh の存在を知らないでよい．

6.2.2 使用するツール

またWorkstation オブジェクトが応用プログラムから読みだされた（アクティベイトされた）ときにRefresh オブジェクトもアクティベイトする処理には、ONTOS が提供する機能を利用できる．また，aRefresh は複数のaWorkstation を管理するためのテーブルを必要とするが，そのテーブルの実現にもONTOS が提供する機能を利用できる．どちらも実現は難しくない．

refresh_interrupt() の登録にはXView のnotify_set_itimer_func() が利用できる．現在ログイン中のユーザを調べるには，RPC（リモートプロシジャコール）やリモートシェルが利用できる．

```
main(int argc, char **argv)
{
 // DBオープン
 OC_open("fukuda_DB");
 // トランザクション開始
 OC_transactionStart();

 // XView 初期化
 xv_init(XV_INIT_ARGS, argc, argv, 0);
 .........
 PopupForWS *aPopupForWS = new PopupForWS(WSName);

 // XView メインループ
 xv_main_loop(aPopupForWS->accessFrame());
 .........
}

PopupForWS::PopupForWS(char *theWorkstation)
{
 .........
 aWorkstation = (Workstation*)OC_lookup(theWorkstation);
 aWorkstation->connectPopupForWS(this);
}

void Refresh::onTime()
{
 ((Workstation *) aWorkstation.Binding(this))
        ->refreshAutonomous();
 aPopupForWS->refresh();
}

void PopupForWS::refresh()
{
 char   string[255];

 sprintf(string, "%s on %s.",
        aWorkstation->accessCurrentUsers(),
        aWorkstation->accessName());
 xv_set(message, PANEL_LABEL_STRING, string, 0);
}
```

図9　コーディング例

6.3 コーディング例

スキーマを定義するレベルでは，クラスのどの内部変数とメソッドが自律性と関連があるかについて詳細に知っておく必要がある．すなわち，Refresh クラスや，Workstation クラスのメソッドrealCurrentUsersのコーディングである．これらのコーディングは注意深く行われねばならない．

一方，応用プログラムを書くレベルでは，OODBの普通のオブジェクトにアクセスするようにプログラムを書くことができる．図9にPopupForWS クラスなどのコーディング例を示す．コンストラクタPopupForWSはWorkstation オブジェクトを読みだし，自分自身との関係を設定している．セレクタrefresh はWorkstation オブジェクトにcurrentUsers の値を尋ね

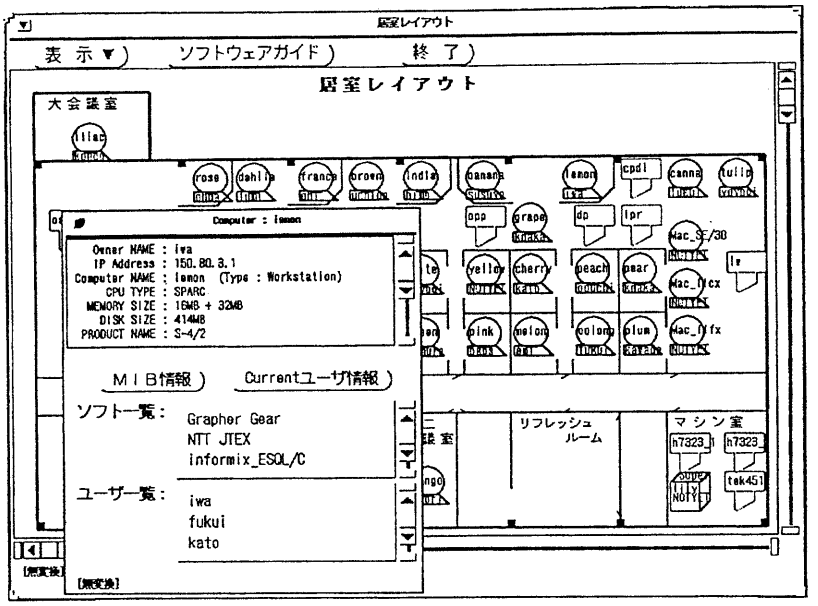


図１０　画面例

て，表示を更新するだけである．すなわち，refresh メッセージの送り手はRefresh オブジェクトであるにもかかわらず，PopupForWS オブジェクトは Workstation オブジェクトからか，main() から送られているように見なしてよい．

図10 にプロトタイプシステムLANDB の画面例を示す．ベースウィンドウには居室のレイアウトが描かれ，ワークステーションが配置されている．ワークステーションをマウスにより選択することで，そのワークステーションに関するデータを表示するポップアップウィンドウが開く．ポップアップウィンドウ内のMIB 情報ボタンやCurrentユーザ情報ボタンを押すと，その時点での情報を表示するポップアップウィンドウが表示される．

7. おわりに

ネットワーク環境における動的データをデータベースへ取り込む問題を解決するために，オブジェクト指向データベースにおける自律オブジェクトのモデルを作成した．また，モデルに基づいて自律オブジェクトを試作した．

本モデルにより，動的，静的の区別することなしにクラスを定義することができるようになり，より自然な形でネットワーク環境のモデル化を行えるようになった．

更に動的データの取り扱いにおいては，単にオブジェクトの内部状態へアクセスするだけではなく，実世界とのデータのやり取りが必要である．実世界とのデータのやり取りをおこなうロジックをオブジェクト指向データベースへ取り込むことにより，応用プログラム群でのロジックの共有が計られ，応用プログラムに対してモデルに基づいた実世界との良好なインタフェースを提供できるようになった．

オブジェクトの自律性をデータベース管理システムに組み込むことにより，動的データをも取り扱えるデータベースへ拡張でき，ネットワーク環境のみでなく，在席管理システムなどをOODB管理システムを中心とした情報システムとして構築できる．

課題としては，ネットワーク管理（特に構成管理）におけるデータベースの役割についての検討がある．現段階では，実世界のネットワークシステムと，データベースを核とした情報システムがダイナミックに一致しているようにすることが必要だと考えている．データベース内のオブジェクトの生成・削除に対応づけられるような，実世界の変化をどのように自律的に取り込んでいくかが問題となる．例えば，LAN では，新規のワークステーションがユーザの手によってネットワークに接続されていき，

ネットワーク管理者がネットワークの全体を把握しにくくなっている．あるいは，そのような情報の管理が煩雑で面倒になってきている．ワークステーションの新規追加に対応して，オブジェクトが生成され，それが自律的にデータベースへ登録されるようなモデルと方式を検討中である．

謝辞

本研究に対し，有益なご助言をいただいた東北大学の野口正一教授，白鳥則郎教授，AICの緒方秀夫常務に深謝いたします．また，システムの構築を支援してくださった富士通BSCの佐藤一浩氏に感謝いたします．また，試作のためにSNMP モジュールを提供してくれたAIC の村田眞人氏に感謝いたします．

参考文献

[1] 福田健一,村田美恵,吉村晋,村田眞人,布川博士,増永良文:ネットワーク環境のオブジェクト指向データモデリングの試み,情報処理学会データベース・システム研究会資料,86-12(1991).

[2] 吉村晋,村田美恵,福田健一,村田眞人,布川博士,増永良文:ネットワーク環境のOODBにおけるヒューマンインタフェースアーキテクチャについて,情報処理学会データベース・システム研究会資料,86-13(1991).

[3] 村田美恵,吉村晋,福田健一,村田眞人,布川博士,増永良文:ネットワーク環境のOODBを用いたモデリングー基本機能ー,情報処理学会第43回全国大会予稿集,5M-5 (1991).

[4] 村田美恵,吉村晋,福田健一,村田眞人,布川博士,増永良文:ネットワーク環境のOODBを用いたモデリングーデータとビューの統合ー,情報処理学会第44回全国大会予稿集,2H-5 (1992).

[5] 福田健一,村田美恵,吉村晋,村田眞人,布川博士,増永良文:ネットワーク環境のOODBを用いたモデリングー自律的データの取り込みー,情報処理学会第44回全国大会予稿集,2H-6 (1992).

[6] 吉村晋,村田美恵,福田健一,村田眞人,布川博士,増永良文:ネットワーク環境のOODBを用いたモデリングー知的質問機能に関する考察ー,情報処理学会第44回全国大会予稿集,2H-7 (1992).

[7] Coad,P. and Yourdon,E.:Object-Oriented Analysis,Prentice Hall(1991).

[8] 増永良文,田中克己:次世代データベースシステムの展望,情報処理,Vol.32,No.5(1991)pp.602-613 .

[9] Taylar,David A.:"Object-Oriented Technology:A Manager's Guide",Servio(1990).

[10] Chakravarthy,S:Rule Management and Evaluation: An Active DBMS Perspective, SIGMOD, Vol.18, No.3(1989) pp.20-28.

[11] Kim, W. and Lochovsky, F.H.:Object-Oriented Concepts, Databases and Applications,acm PRESS(1989).

[12] 江允,牧之内顕文:並列オブジェクト指向永続プログラミング言語WARASAの並列機能,情報処理学会第43回全国大会予稿集,4M-11(1991).

[13] Ontologic Inc.:Ontos Object Database Developer's Guide (1991).

[14] XView Programing Manual(1991).

[15] 1990年度WIDE 報告書(1991)pp.66-79.

[16] RFC1155,1156,1157,1158.