社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－124（4）
2005－ GN － 57（4）
2005／9／21

ネットワークプログラミング向けスコープ概念の実装と評価

鴨川 雄† 由井薗 隆也††
†島根大学大学院総合理工学研究科 ††島根大学総合理工学部
E-Mail : { s049304 , yuizono } @cis.shimane-u.ac.jp

ネットワークプログラミングは，主に専用の API 群を用いて行われる．この場合，コード記述量が増加しプログラムの可読性が下がる可能性がある．そこで，プログラミングにおける領域概念を拡張し，ネットワーク上に存在する資源の操作をローカル上に存在する資源の操作と同様の記述を用いて実現させる方法 EXDNA について提案する．これにより，ネットワークプログラミングにおけるコード記述量の減少とプログラムの可読性の向上を狙う．

Implementation and Evaluation of a Concept of Scope for Network Programming

Takeshi Kamogawa† Takaya Yuizono††
† Graduate School of Science and Engineering , Shimane University
†† Faculty of Science and Engineering , Shimane University

A specific APIs is usually used for developing a network program. In the case, there is a possibility to make size of code descriptions up and program's readability down. Then, we propose EXDNA, which is a method of letting programmers handle resources on network in the same way as local ones by extending concept of area in programming. By using EXDNA, we aim at reducing size of code descriptions and at improving program's readability of network programming.

１．まえがき

近年，計算機の低価格化やマイクロプロセッサとネットワーク環境の発達により，複数の計算機群を用いて高度なサービスを提供する分散処理システムが多く開発されるようになった．今後，このようなシステム開発のためのソフトウェア開発支援環境はますます重要になる．

現在の分散処理システムの開発[1]では，ソケットなどの低レベルの API(Application Programming Interface) や手続き概念などを用いて抽象化記述が行える RPC(Remote Procedure Call)，分散オブジェクト環境下で遠隔上に配置したオブジェクトへの操作インタフェースを提供する HORB(Hirano Object Request Broker)[2]や RMI(Remote Method Invocation)[3]などが用いられる．これらのものを用いることで，プログラマはネットワークをあまり意識することなくシステムを記述することが可能になる．しかし，依然として通信接点の管理や遠隔上のオブジェクトに対応する参照体の取得など，ネットワーク処理に特化された記述が存在する．また，システムを構成する各プログラムのシステム間での統合を開発者側の責任で行わなくてはならないなど，改善の余地が多く残されている．

本報告では，既存の高級言語を拡張しネットワーク機能を与える方法 EXDNA(Extended-one for Developing Network Application)について提案し，実際にこの考えをプログラミング言語の１つである Java[4]に適用した言語 Java-EXDNA の説明とその評価について述べる．

２．ネットワークプログラミングに対する考察

ネットワークプログラミングを特徴付ける重要な要素は，大きく分けて遠隔上に存在するサービスとそのサービスを受けるためのインタフェースの２つである．ソケットなどの低レベルの API で複雑なデータを送受信する場合には，始めに API 固有の知識を修得し，アプリケーション固有のプロトコル等を設計しなくてはならない．この設計を一から行った場合，開発者側は大きな負担を強いられる．また，プログラム全体の内容を理解するためには，複数のプログラムに渡って目を通す必要がある．そのため，一般に分散処理システムの開発全般は単一計算機での動作を想定したシステムよりも困難である．

一方で HORB や RMI では，プログラマが定義した遠隔上に設置するオブジェクトの定義情報から，それにアクセスするためのモジュールとそれを管理するモジュールを（場合によってはその実体も）生成し開発者側に提供する．これにより，開発者側は複雑なソケットの操作やアプリケーション固有のプロトコル等の設計をする必要がなくなる．また，オブジェクトごとに ID を与えることができるため，シ

ステム内においてその存在を明確にすることができ，プログラムは従来のものよりいくらか読み易くなる．しかし，モジュールの使用については開発者側が手作業でシステムに組込まなくてはならないため，誤った使用により潜在的な欠陥を生じさせる可能性がある．このことを逆の立場から考えた場合，モジュールを使用すべき場所やタイミングが明らかであるなら，システム中の適切な箇所への組込みを一定の規則に従って機械的に行うことも可能である．

また，一般的なコンパイラにおける翻訳単位[5]は，１モジュールまたは１プログラム単位である．そのため，分散処理システムの開発では一つずつ各構成プログラムをコンパイルした後に全体を統合する．そのため，統合して初めて不具合が見付かる場合もある．しかし，翻訳単位を複数のプログラム単位まで拡張できれば，システム全体の処理に関する情報の収集に加えて，明らかな間違いや実行時において想定される不具合の発見が期待できる．

以上のことから，現状のネットワークプログラミングの開発環境においても，機械的な支援が行える箇所がまだ多く残されていると言える．

３． EXDNA

EXDNA では，既存の高級言語を拡張しネットワーク機能を与える．具体的には，コンパイラ技術を応用して以下に示す３つを実現する．

（1） 外部のプログラムからアクセス可能な公開性の高い資源の提供
（2） 外部のプログラムが所有する資源に対する単純な操作インタフェースの提供
（3） ネットワーク上に存在するプログラムの明確な位置の特定

本研究では，既存のプログラミング言語の１つである Java に対して EXDNA の考えを適用した言語を開発し，その有効性について検証した．以降では，EXDNA における各アイディアの説明に加えて，Java におけるその実装例についても説明する．

３．１ 基本アイディア

一般に，コンピュータプログラムは大きく分けて変数と関数の２つの資源から構成される．変数の役割はデータの記憶である．また，関数の役割は与えられたデータに対するまとまった一連の処理を行うことと，処理されたデータの呼出元への返却である．これらのものは，それ自身が宣言される領域によって与えられる性質が違ってくる．

例えば，「局所領域」に宣言された場合，その資源はプログラムを構成する一部のモジュールからのみアクセス可能となる．またその資源が変数である場合，その実体はそれが必要な時に生成され，不要になった時点で消滅する．一方で「大域領域」に宣言された場合，その資源はプログラムを構成する全モジュールからアクセス可能となる．またその資源が変数である場合，その実体はプログラムの実行開始から終了まで常に存在し続ける．

EXDNA では，コンピュータプログラムに対して大域領域を拡張した領域概念と，その領域上に宣言された資源に対する単純な操作インタフェースを導入することでネットワークプログラミングの開発支援を行う．この新たに導入される領域のことを「ネットワーク公開領域(Public Areas on Network : PAN , ネット領域)」[6,7,8]と呼ぶ．また，この領域上に宣言された資源のことを「ネットワーク公開資源（ネット資源)」と呼ぶ．

３．２ プログラムの基本情報

EXDNA において，プログラムはそれぞれ固有の識別子と動作する計算機の IP アドレス，データ通信用のポート番号に関する情報を持つ． また，特定のプログラム群で集合を形成することも可能である．集合の情報は，集合の識別子と集合に属する各プログラムの情報からなる．これらの情報は，専用のファイルに定義し，ネットワーク上に存在する各プログラムの位置を特定するために用いられる．このファイルを「ネットワークファイル」と呼ぶ．このファイルには，個々のプログラムをコンパイルする際に用いるオプションの情報も定義する．

図１は，EXDNA におけるシステムのイメージである．また図２は，図１のシステムを定義するためのネットワークファイルの記述例である．

３．３ ネットワーク公開領域

ネットワーク公開領域の特徴は，領域上に属する資源に与えられる高い公開性と長い寿命，資源に対する単純な操作インタフェースの提供である．具体的には，この領域上に属する資源はプログラムの実行開始から終了まで常に実体が存在する．また，資源を所有するプログラムを構成する全モジュールに加えて，外部に存在するプログラムを構成する全モジュールからもアクセス可能となる．

図３は，図１のシステム中に存在するプログラム Cerberos が持つ各領域に対する，外部のプログラムからのアクセス性を表わしたものである．

ネット領域への資源の宣言は，特定のファイルに行う．このファイルを「リソースファイル」と呼ぶ．資源の宣言は，IDL(Interactive Data Language) [9]

を用いて行うのではなく，適用された言語が持つ一般的な資源の宣言方法を用いて行う．この領域に宣言できる資源は変数と関数の２つである．ネット資源である変数のことを「ネット変数」，ネット資源である関数のことを「ネット関数」と呼ぶ．使用可能なデータ型は，原則として適用された言語が持つ全ての型（ユーザ定義型も含む）とする．

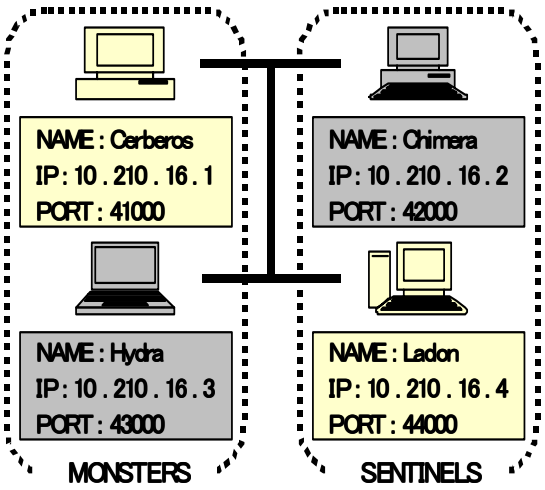

図１ EXDNA におけるシステムのイメージ

```
#PROGRAM Cerberos 10.210.16.1 41000
#C-OPTION –classpath c:¥lib¥material-C.jar;
#PROGRAM Chimera 10.210.16.2 42000
#PROGRAM Hydra 10.210.16.3 43000
#C-OPTION –classpath c:¥lib¥material-H.jar
#PROGRAM Ladon 10.210.16.4 44000

#GROUP MONSTERS    Cerberos    Hydra
#GROUP SENTINELS    Chimera    Ladon
```

図２ ネットワークファイルの記述例

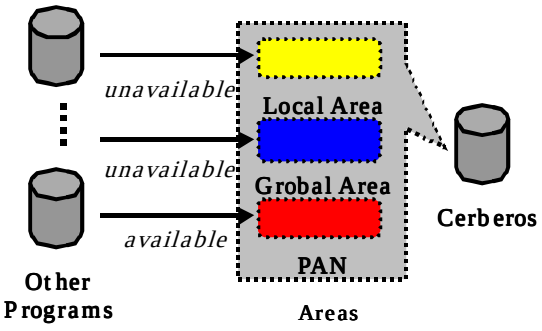

図３ 各領域に対する外部からのアクセス性

Java-EXDNA におけるリソースファイルは，各プログラムに与えられた識別子と同じ識別子を持つクラス内に宣言された資源の一部がネット資源として認識される．ただし，このクラスには次に示す７つの意味的な制約が課される．

（１） クラスの属性は public であり abstract
（２） いかなるクラスも継承不可能
（３） いかなるインタフェースも実装不可能
（４） いかなるクラスへの継承不可能
（５） 内部クラスや内部インタフェースは所有不可能
（６） 所有する資源の記憶クラスは全て static
（７） アクセス属性が public であり性質が final であるものは所有不可能

以上の条件を満たしたクラスが所有する資源の中で，アクセス属性が public であるものがネット資源として認識される．また使用可能なデータ型は，直列化可能なデータ型[4]全てである．図４は，ネットワーク上に存在するプログラム Cerberos におけるリソースファイルの記述例である．

```
public abstract Cerberos {
    public static int V ;
    public static int F ( int x , int y ) {
        return ( x + y ) ;
    }
    public static String   MSG ;
}
```

図４ リソースファイルの記述例

３．４ ネットワーク公開資源

EXDNA において，各プログラムから外部に存在するプログラムが所有する資源の中でアクセス可能なものは，原則としてネット資源のみである．ネット資源の操作は，専用の関数や参照体，構文規則を用いて行うのではなく，操作対象のネット資源に対応した専用の識別子と，EXDNA が適用された言語が持つ一般的な構文規則を用いて，ローカル上に存在する資源の操作と同様の処理記述を用いて行う．

以下に，ネット資源を指す識別子の定義を示す．個々のプログラムが所有するネット資源を指す識別子は，資源を所有する「プログラムの識別子」と「資源の識別子」を“．”（ピリオド）で繋いだものである．また，特定のプログラム群が個々に持つ定義情報が同じネット資源を指す識別子は，「集合の識別子」と「資源の識別子」を“．”で繋いだものである．

```
＜ネット資源の識別子＞
::= ＜プログラムの識別子＞．＜資源の識別子＞
    |    ＜集合の識別子＞．＜資源の識別子＞
```

図５は，ネットワーク上に存在するプログラム Cerberos が所有するネット変数 V とネット関数 F に対する操作と，集合 MONSTERS に属する個々のプログラムが所有するネット変数 V に対する操作の記述例である．これらは単純な処理だが，ネット資源は原則として EXDNA が適用された言語の文法に違反しない限り無制限に使用できるものとする．

```
……..
// Store a data in a variable V on program “Cerberos”
Cerberos.V = 0x2BADBABE ;
// Load a data variable V on program “Cerberos” to X
X = Cerberos.V ;
// Call a function f on Program “Cerberos”
Y = Cerberos.F( 0xFEED , 0xF00D ) ;
// Store a data in a variable V
// on each program in MONSTERS
MONSTERS . V = 0xDEADBEEF ;
……..
```

図５ ネット資源の操作記述例

３．５ コンパイラ

EXDNA が適用された言語におけるコンパイラ[8]の翻訳単位は，ネットワークファイルに定義された全プログラム単位である．この理由としては，システムを構成する全プログラムを一括してコンパイル処理を行うことに加えて，各プログラムが所有するネット資源の情報やシステム全体の処理の流れに関する情報を収集し，明らかな誤りや実行時に想定される不具合を発見することにある．主な処理内容は，通常のコンパイル処理に加えて以下に示す４つの処理を行うことである．

（１） ネット資源の実体の生成
（２） ネット資源を操作するための命令群を持つクライアントモジュールの生成
（３） ネット資源に対する外部からのアクセスを管理するサーバモジュールの生成
（４） （１）～（３）における生成物のプログラム中の適切な箇所への組込

図６は，EXDNA コンパイラが行う処理のイメージである．先にあげた（１）～（４）の処理は，図中の Preprocess 部分で行われ，通常のコンパイル処理は Compile 部分で行われる．

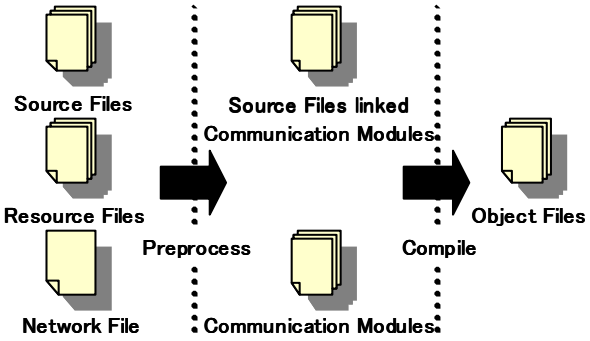

図６ EXDNA コンパイラの処理イメージ

３．６ 生成物の役割と動作

EXDNA コンパイラが生成するクライアントモジュールは，表１にあげる４種類のものを持つ．端末変数とロード命令およびアップデート命令の３つは，ネット変数の情報から生成される．端末変数は，対応するネット変数と同じデータ型の変数であり，ロード命令とアップデート命令の処理の配下に置かれる．ロード命令は，対応するネット変数の値を配下にある端末変数に読込む処理を行う．アップデート命令は対応するネット変数の値を配下にある端末変数の値で書換える処理を行う．また端末関数は，ネット関数の情報から生成される．端末関数では，与えられた引数を用いて対応するネット関数を呼出し，その戻り値を取得して呼出元に返す処理を行う．

表１ ネット資源に対する生成物

生成物	機能の概要
端末変数	対応するネット変数の値の記憶
ロード命令	対応するネット変数の値の取得
アップデート命令	対応するネット変数の値の書換
端末関数	対応するネット関数の遠隔呼出

サーバモジュールは，クライアントモジュールからの命令に従って，管理下にあるネット資源に対する操作を行う．ロード命令に対しては，対応するネット変数を一時的に外部からアクセス不可能な状態にする処理と，ネット変数の値のコピーを返す処理を行う．アップデート命令に対しては，対応するネット変数の値を与えられたデータで書換える処理と，対応するネット変数を外部からアクセス可能な状態にする処理を行う．端末関数に対しては，与えられたデータで対応するネット関数を呼出し，その戻り値を返す処理を行う．

これらのモジュールの役割は，HORB や RMI における Proxy や Stub，Skeleton[2,3]のそれと類似している．また注意点として，モジュール間で送受

信されるデータは，原則として全てオリジナルデータのビットコピーとする．そのため，ポインタ値を含む構造体の場合その取り扱いはプログラマ側の責任に委ねられる．

３．７ ネット領域の形成

EXDNA コンパイラは，翻訳対象のソースファイルをコンパイル前に再編集し，生成されたクライアントモジュールとサーバモジュールを組込む．この処理により，各プログラムが所有するネット資源をプログラム間でリンクすることでネット領域が形成され，各プログラムは外部のプログラムが所有するネット資源にアクセス可能な状態になる．

プログラムの開始部分には，クライアントモジュールとサーバモジュールを起動させる命令を組込み，プログラムの終了部分には，クライアントモジュールとサーバモジュールを停止させる命令を組込む．

個々のプログラムが所有するネット変数を操作する処理記述に対しては，その前後にロード命令とアップデート命令を挿入し，記述中にあるネット変数の識別子を対応する端末変数の識別子に書換える処理を行う．ネット関数を呼出す処理記述に対しては，記述中にあるネット関数の識別子を対応する端末関数の識別子に書換える処理を行う．

また集合に対する操作処理記述に対しては，集合の識別子を集合に属する個々のプログラムの識別子に展開後，個々のプログラムが所有するネット資源を操作する処理記述に対して，先にあげた処理と同様のことを行う．

図８は，プログラム中でネット変数を操作する処理記述とネット関数を呼出す処理記述を見つけた場合に行われる処理のイメージである．

４． システム評価

４．１ 実験内容

評価実験では，数値計算（円周率），画像処理（マンデルブロー集合），チャット，スロットマシンの４つのプログラムをそれぞれEXDNAとHORB, RMIを用いて開発し，それらのコード記述量（行数）を比較した．これら３つは，全て Java をベースにしたものである．また，与えられた整数３つの合計値を返す単純な処理を行う関数をネットワーク上に配置し，EXDNA と HORB，RMI で開発されたそれぞれのプログラムの応答時間を計測した．

４．２ 実験環境

実験環境は，ネットワークが 100Mbps の LAN で，使用した計算機のスペックは OS が MacOS X Ver. 10.4 , CPU が PowerPC G4 700MHz，メモリが SDRAM 512MB である．また，使用した Java と HORB はそれぞれVer．1.4 とVer．2.0 である．

```
.....
Cerberos.V = 0x2BADBABE ;  // A network variable operated .
.....
Y = Cerberos.F ( 0xFEED,0xF00D ) ; // A network function called .
.....
// Network variables in MONSTERS operated .
MONSTERS.V = 0xDEADBEEF ;
.....
```

Original Codes

```
.....
Cerberos.Load$V ( ) ; // Load command inserted .
Cerberos.$V = 0x2BADBABE ; // Variable identifier exchanged .
Cerberos.Update$V ( ) ; // Update command inserted .
.....
Y = Cerberos.$F ( 0xFEED,0xF00D ) ; // Function identifier exchanged .
.....
// Group identifier extended  to each program's one in MONSTERS .
// Each program's variable identifier exchanged .
// Load commnads and update ones inserted .
Cerberos.Load$V ( ) ;
Cerberos.$V = 0xDEADBEEF ;
Cerberos.Update$V ( ) ;
Hydra.Load$V ( ) ;
Hydra.$V = 0xDEADBEEF ;
Hydra.Update$V ( ) ;
.....
```

Preprocessed Codes

図８ モジュールの組込み

４．３ 実験結果

以下に，システム評価で開発された各プログラムのコード記述量と，ネットワーク上に配置した関数の呼出に対する各平均応答時間を載せる．

表２ コード記述量（行数）

プログラム	EXDNA	HORB	RMI
数値計算	323	545	756
画像処理	317	415	543
チャット	97	132	169
スロットマシン	199	222	245

表３ 平均応答時間（ms）

	EXDNA	HORB	RMI
応答時間	17.3	0.6	0.7

５． 考察

５．１ コード記述量の評価

表２より，プログラムのソースコードの大きさは EXDNA により開発されたものが一番小さい．この理由として，HORB や RMI ではプログラムのソースコードに通信環境の初期化やデータ通信処理に特化された処理記述が存在することに加えて，RMI ではさらに外部との通信処理に対する例外の処理記述が存在

することにあると考えられる．

また遠隔上の変数資源に対して操作をする場合，EXDNA では直接的な処理記述により操作が可能だが，HORB や RMI では各変数に対して専用の関数（データの読込や書換など）を用意する必要がある分，EXDNA と比較してコード記述量が大きくなる．

５．２ 通信速度の評価

表３より，遠隔上に存在する関数の平均応答時間は EXDNA が一番遅い．EXDNA におけるデータ通信処理では，Java 言語が標準で持つソケット通信のライブラリとデータ通信用の直列化機能を組合せた独自のものが用いられている．今回のように極端な差が出た理由が，単に通信モジュール部分における問題であるならば，モジュールの改良または交換により性能の改善が見込める．

５．３ 言語の評価

Java を拡張しネットワーク機能を与えた言語に，mJava/LR[10]と MetaVM[11]がある．これらの言語には，オリジナルと比較して新たに追加された予約語や構文規則があるため，基本的には Java とは別の言語である．一方 Java-EXDNA では，ネットワークファイルの導入とネット領域に関連した意味規則の追加がされたものの，予約語や構文規則に関する変更が一切ないため，Java のスーパーセットである．

Java への EXDNA の適用結果から，EXDNA は適用対象の言語にもよるが，Java 言語と類似したオブジェクト指向言語や手続き型言語（C や C++[12]など）に関しては，オリジナルとの差がほとんどない状態で適用できる可能性がある．

５．４ プログラムの性質

今回開発した EXDNA コンパイラは，静的なネットワークシステムの開発向けに設計されているため，プログラムの基本情報と集合の情報はコンパイル処理時にしか利用されない．そのため，システム実行中におけるプログラムの追加や削除，プログラムの実行環境の変更，集合を構成するプログラムの変更，新しい集合の追加や削除などの動的な変化に対する処理がサポートされていない．

今後は，本開発環境を動的なネットワークシステム開発にも適用できるように，先にあげたような動的な変化を扱う機構を検討していく予定である．

６．まとめ

本研究では，既存の高級言語を拡張しネットワーク機能を与える方法 EXDNA について提案した．

EXDNA では，プログラミングにおいてネットワーク上に存在する資源を指す一意な識別子を導入する．プログラマは，この識別子を用いることでネットワーク上に存在する資源に対する直接的な操作処理記述が可能となる．またコンパイラにおいては，ソースコードを解析し，ネットワーク上に存在する資源を操作するための具体的な命令を自動生成する．これをソースファイル中の適切な箇所に組込む処理を行い，プログラムにネットワーク機能を与える．このアイディアをプログラミング言語の１つである Java に適用したところ，従来のものを用いて開発されたプログラムと比較して，少ない記述量で同等のプログラムを開発することに成功した．

今後は，コンパイラ部分だけでなく，プログラムの実行時において想定される問題も視野に入れ，それを解決するためのアイディアの検討を中心に研究を進める．また，第三者による EXDNA のプログラミングの評価も行う予定である．

参考文献

[1] A. S. Tannenbaum , M. V. Steen : 分散システム 原理とパラダイム , Pearson Education Japan , 2003.

[2] HORB : http://horb.etl.go.jp/horb-j

[3] Sun Microsystems : Java Remote Method Invocation Specification JDK 1.2 , 1998.

[4] K. Arnold , J. Gosling , D. Holmes : プログラミング言語Java第３版 , ピアソン･エデュケーション , 2001.

[5] 中田 育男 : コンパイラの構成と最適化 , 朝倉書店 , 1999.

[6] 鴨川 雄 , 由井薗 隆也 : 分散処理プログラム作成支援ツール PMN の提案 , 第５回 IEEE 広島支部学生シンポジウム論文集 , pp.207-210 (2003)

[7] 鴨川 雄 , 由井薗 隆也 : 分散処理プログラム開発用言語 JavaPAN の開発と評価 , 第６回 IEEE 広島支部学生シンポジウム論文集 , pp.239-242 (2004)

[8] 鴨川 雄 , 由井薗 隆也 : ネットワーク上に静的な領域を持つプログラミング言語 JavaPAN の提案 , 電子情報通信学会論文誌 , VOL.J88-D-1 , NO.2 , pp.326-329 (2005)

[9] CORBA : http://www.corba.org/

[10] 渡辺 昌寛 , 伊藤 貴康 : 場所の概念を備えた Java 言語とその処理系 , 情報処理学会論文誌 , Vol.40 , No.SIG7 (PRO4) , pp.51-65 (1999)

[11] 首藤 一幸 , 根山 亮 , 村岡 洋一 : プログラマに単一マシンビューを提供する分散オブジェクトシステムの実現 , 情報処理学会論文誌 , Vol.40 , No.SIG7(PRO4) , pp.66-79 (1999)

[12] Bjarne Stroustrup : プログラミング言語 C++ 第３版 , Addison-Wesley Publishers Japan , 1998.