# 分散環境のための言語系 DeLis

三石大[1] 布川博士[2] 宮崎正俊[3] 野口正一[1]
1. 東北大学応用情報学研究センター 2. 東北大学電気通信研究所 3. 東北大学教養部

大規模分散システムの個人レベルでの利用機会が増え、より高度なサービスのあり方が求められている。

本稿では、分散システム上の各種サービスをその利用者にとって環境として提供するために、「分散環境 (Decentralized Environment)」を提案し、この分散環境を分散システム上へ構築するための言語系 DeLis の設計、実装について述べる。

我々が実装した DeLis は Lisp をベースとしており、プログラム (関数) を一階のオブジェクトとして扱うことができる。また、通信や GUI の機能も関数として記述することができ、分散システム上に柔軟なシステムとして分散環境を記述できるという特徴を持つ。

# Programming Language DeLis for Decentralized Environment

Takashi Mitsuishi*, Hiroshi Nunokawa†, Masatoshi Miyazaki‡ and Shoichi Noguchi*
*Research Center for Applied Information Sciences, Tohoku University
†Research Institute of Electrical Communication, Tohoku University
‡College of General Education, Tohoku University

As opportunities to use a large scale distributed system in a private level go on increasing, more advanced services are being required.

In this article we propose a "Decentralized Environment" in order to provide all sorts of services as environments for users over a distributed system. We also explain the design and implementation of the DeLis — a language system for constructing the decentralized environment over a distributed system.

DeLis, which we implemented, is based on Lisp. It is able to both handle a program (function) as a first class object and describe communication and GUI functions. The characteristic of DeLis is that we can describe decentralized environment as a flexible system over a distributed system.

# 1 はじめに

## 1.1 背景

近年、コンピュータの性能の飛躍的な向上と共に、その相互接続によるコンピュータネットワークが普及し、それらがさらに接続され世界的レベルでのインターネットが形成されてきている。そして、この様な大規模分散システムへの個人でのアクセスの機会が増えてきている。

ここでいう分散システムとは、コンピュータネットワークすなわちコンピュータ同士が相互に接続された物理的なシステムのことであり、この分散システム上にはメイルやニュースなどといった様々なサービスが提供されている。個人利用者は、この分散システム上に提供される各種サービスの利用を目的としていることが多い。

現在このような分散システムではコンピュータの端末上で分散システムに対し処理要求を出すことによりのみそのサービスを受けることができ、また、そのサービス提供の形態は、コンピュータネットワークに常に依存し、ユーザはそれを意識し利用しなければならない。

しかしながら一般のエンドユーザ (以下ユーザとはこの様な利用者を指す) の要求はコンピュータそのものの利用ではなく、その上で実現される高度なコミュニケーションにある。また、コンピュータの初心者、非専門家にとってはこれらのサービスの利用は複雑でわかりづらいものとなっている。

その結果、このような分散システム上で提供されるサービスは便利なものが多いにも関わらず、一般の利用者にはなかなか馴染めず、また充分な利用が行なえないのが現状である。

また、現在提供されているサービスのほかに、このコンピュータネットワークという分散システムはさらに多くの新しいサービスを提供できる可能性をもち、それを提案していくことができる。

## 1.2 分散環境

分散環境とは、分散システム上に実現される各種サービスを環境としてユーザに提供するものである。各種サービスとは、先に挙げたメイルやニュース、グループウェアといったものであり、これらのサービスの中には既に提供されているものもあるし、これから提供されていくであろうものもある。

すなわちこの様なサービスを初心者から熟練者まで、ユーザが“利用できる”形で提供することが必要であり、そのユーザを取り巻く環境として機能させることが必要であると我々は考える。これが分散環境である。例えば DoReMi[5] は都市メタファを用いることにより、仮想的な都市利用として環境を提供している。

ユーザが分散システムを利用する場合、コンピュータやネットワークという物理的なシステムは意味があるものではない。分散システム上のサービスをコンピュータやネットワークというものを意識させず、そのサービスの意味を理解させることが必要である。そのためには何らかのユーザインタフェースを通し提供することが必要となる。また、これらのサービスは相互に組み合わせることによりより利用価値がある。熟練者はコンピュータ上でシェルなどを利用しサービスの組合せを行なうことができるが、これは初心者には難しい。したがってこれらサービスを統合し、提供することが必要である。すなわち図1のような形でユーザに提供される必要がある。

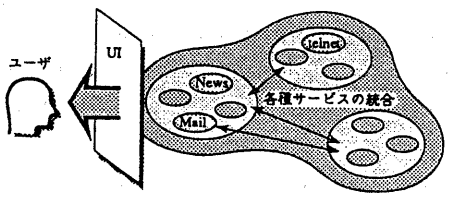


図 1: サービスの提供

### 1.3 目的

本稿の目的は分散環境の必要性とその構成形態を明かにし、これに基づき分散環境の構築手法を考察すること、そして、分散環境のためのプログラミング言語を提案し、その設計、実装を行い、実際に分散環境のためのアプリケーションを記述することによりその有効性を示す。

### 1.4 本論文の構成

本論文では、第2章で分散環境について述べその構成を明らかにするとともに、それを実現するための言語系の仕様を考える。第3章では、その言語仕様に基づき設計、実装されたDeLisの特徴を述べDeLisによる分散環境の構築法について述べる。さらに第4章では、DeLisを用いて作成されたアプリケーションの例を示し、それによるDeLisの評価を行なう。最後に第5章でDeLisについて考察を行ない、今後の課題について検討を行なう。

## 2 分散環境

### 2.1 分散環境の構成

分散環境は、分散システム上に実現される各種サービスから得られる情報を統合し、何らかの加工を行いユーザに提供するものである。これは、次のように定義することができる。

分散環境は、いくつかの情報ユニットとその間でのコミュニケーションから構成される。また、分散環境は分散システムやユーザとのインタフェースを有する。

情報ユニットとはサービスを提供する主体であり、分散環境内の情報の単位である。この情報ユニットが互いにコミュニケーションを行ない情報を交換することによりサービスを統合、加工することができる。

また、分散環境はそれ自身のみで存在しても意味がなく、分散システム上のサービスやユーザとの対話を行わなければならない。これは以下のような理由による。

情報ユニットは、それ自身が情報を生み出すことも可能であるが、多くは分散システム上のサービスを取り入れそれを情報として発生させる。そして、これらの情報を分散システム上のサービスを利用して加工する場合もある。すなわち分散環境は、分散システムとの情報の交換を行うためのインタフェースを有していなければならない。

分散環境内で発生、加工された情報は、さらにユーザに提供され、また分散環境はユーザからの情報を受けとりそれにしたがって処理を行なうことが必要である。すなわち、それらの情報をユーザに提供し、ユーザからの情報を入力するためのインタフェースを有していなければならない。

これはすなわち図2の様に表すことができる。

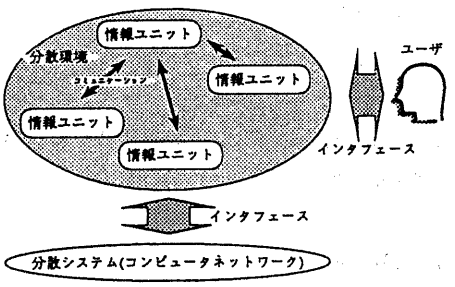


図 2: 分散環境

分散環境の構成は以上のように表すことができるが、その形態は固定ではない。ユーザの要求は分散システムの利用と共に拡大し、サービスについても、変更や新たなサービスの追加が予想される。すなわちユーザからの要求や、分散システムは絶えず変化する。分散環境の構成形態もこの変化に合わせていかなければならない。すなわち、分散環境の構成形態は動的に可変であることが必要である。

### 2.2 分散環境のための言語系

この様な分散環境を構築しようとするわけであるが、しかしながら、現在のところこのための手段はない。また、はじめに分散環境を記述

すると同時に、分散環境の動的可変性を実現するためにその変化する分散環境を維持、管理することが必要である。そこで我々はそのための言語系を設計した。

先に述べたが、分散環境は情報ユニットとその間のコミュニケーションから構成され、分散システムやユーザ等外界とのインタフェースを有する。そしてその形態は絶えず変化することが予想される。また、より多くのシステム上で実現されることが望まれる。従ってこの分散環境のための言語系に要求される言語仕様は大きく分けて

1. 情報ユニットの記述
2. 情報ユニット間コミュニケーションの記述
3. 外界とのインタフェースの記述
4. 動的可変性
5. 実装の容易性
6. 拡張性

の6つである。

これらの要求を比較的満たしているものとして我々はLispを選択し、これを拡張することにより分散環境記述のための言語を作成した。

## 3 DeLis

### 3.1 DeLisの特徴

DeLis(Decentralized Lisp Interpriters)は分散環境を構築し運営していくための言語系として設計されたものであり、現在UNIX上で実現され、以下のような特徴を持っている。

#### 3.1.1 Lispとしての DeLis

DeLisは基本的にはLispインタプリタの集合であり、個々のDeLisはそのLispとしての特徴を持っている。

DeLisは関数的言語であり、プログラムは関数の集合として定義される。関数は、プログラム内で新たに定義し使用することができる。そのプログラムはS式の形で記述される。記述されたプログラムはインタプリタによって実行される。また、扱われるデータもS式で表現される。プログラム自身をデータとして扱うことも可能である。

ここでいう関数とは拡張された意味での関数であり入力に対し系に何らかの変化を与えることが可能であり、状態に応じた結果を返す。

Lispインタプリタは比較的単純な構造をとり、またそこで扱われるS式もポインタを用いて簡単に扱うことができる。したがって、C言語等により容易に実現することが可能であり新たな機能の追加等の拡張性に富む。

また、プログラムをデータとして扱いその中で新たな関数を定義でき、それをインタプリタにより実行することで動的可変性を実現できる。

#### 3.1.2 通信機能

DeLisはDeLis-DeLis間通信のための機能を有している。

DeLisは分散システム上に複数存在し、各々が独立に動作することが可能である。そして、この通信機能を利用してその間でのコミュニケーションを行なう。

これは、CSP(Communicating Sequential Processes) [3]に似た以下のような関数により実現される。

(open)
(connect *host-name*)
(send *channel-number*)
(recv *channel-number*)
(alt *channel-1 channel-2* ...)

すなわちチャネルを介した1対1同期通信である。しかし、CSPとは異なり動的なチャネルの生成破棄が行なえる。

この通信機能により、S式の送受信を行なうことができるのであるが、その上でコミュニケーションを行なうためのプロトコル、すなわちコミュニケーションプロトコルはS式を用いてプログラム内で関数として定義することができる。したがってLispであることにより、

プロトコルの動的生成

プロトコルの動的変更

が可能となっている。現在この通信のための機能は、UNIX(BSD系)で提供されるTCP/IP上のソケットを使用することによって実現されている。

### 3.1.3 シェル機能

DeLisは以下のようなシェルの機能を有している。

(exec *command-name argument string*)

この関数は分散システム上のコマンドを実行するためのものであり、引数をそのコマンドへ渡し、コマンドからの標準出力をこの関数の評価値とする。

この関数により分散システムに対して直接制御を行なうことができ、また、分散システムとのデータの交換を行なうことが可能となる。例えばmailやtelnet等のコマンドを実行することにより、分散システムに対してデータを出力したり、また入力を行なうわけである。

### 3.1.4 高レベルユーザインタフェース部品

DeLisは以下のような関数を用意することによりグラフィカルユーザインタフェースの機能を有している。

(create-window ...)

(text-field ...)

(get-text ...)

(make-button ...)

(get-event ...)

これらの関数を利用し、組み合わせることによりテキストの入出力を行なうためのフィールドやボタンなどを画面に出力し、マウスやキーボードからのイベントの入力を行なう。

これらの関数は今のところXウィンドウシステムの機能を利用することにより実現されているが、極めて単純かつ高レベルなものであり、MS-Windowsやその他パソコン上で実現されているウィンドウシステムでも簡単に実現できることが予想される。

## 3.2 DeLisによる分散環境の構築

以上のような特徴を有するDeLisであるが、このDeLisを用いて分散環境がどのように構築されるかを示す。

### 3.2.1 情報ユニットの記述

分散環境を構築するには、先ず分散環境の構成単位である情報ユニットを記述する必要がある。この情報ユニットであるが、これはDeLisの関数として定義される。したがって、情報ユニットによる情報の発生はその情報ユニットとして定義されている関数を評価することにより行なわれる。

### 3.2.2 コミュニケーションの記述

情報ユニット間コミュニケーションは基本的にチャネルによる通信機能により行なわれる。

プロトコルは動的に定義、変更することが可能であるので、あるサービスのためにこれから通信をしようとするDeLisどうしが相手の持つコミュニケーションプロトコルを知らなくても、その場で相手のプロトコルに合わせて定義することができる。これは、DeLis同士が協調することを意味する。

### 3.2.3 分散システムとのインタフェースの記述

分散環境と分散システムとのインタフェースにはシェル機能を利用する。分散システム上のコマンドを実行することにより、そのコマンドを通して分散システムとの情報の交換を行なうわけである。

さらに、情報ユニット間コミュニケーションを上記のチャネルによる通信機能を利用する以外で、mailコマンドなどにより分散システム上のサービスを利用することによっても行なうことが可能となる。

### 3.2.4 ユーザインタフェースの記述

分散環境とユーザとのインタフェースにはグラフィカルユーザインタフェースの機能を利用する。

多くの場合、ユーザインタフェースとして必要となるのはテキストの入出力、絵の表示、ボタンからの入力といったものの組合せで実現できることが多い。例えばユーザからの入力を促すためのダイアログなどはテキストの出力とボタンを組み合わせることにより図3のように容易に実現することができる。

```
(defun dialog
  (window string1 string2 string3 x y width height)
  ((lambda (dialog-window half-width half-height)
     ((lambda (button1 button2)
        ((lambda (pushed-button)
           (progn (text-field dialog-window string1
                              10 10 width half-height nil)
                  (cond ((equal pushed-button button1) 1)
                        ((equal pushed-button button2) 2))))
         (get-event button1 button2)))
      (make-button dialog-window string2
                   10 (+ half-height 20)
                   half-width half-height)
      (make-button dialog-window string3
                   (+ half-width 20) (+ half-height 20)
                   half-width half-height)))
   (create-window window x y width height)
   (/ (- width 30) 2)
   (/ (- height 30) 2)))
```

図 3: ダイアログの記述

この様に基本的なユーザインタフェースを組み合わせて新たに定義することにより、様々なユーザインタフェースを実現でき、それぞれのサービスに対してそれに応じた適切なインタフェースを提供することができる。

また、これらの関数が実現されれば、UNIX ワークステーション以外に例えばビットマップディスプレイを持つパソコンへも、容易に同じインタラクションを行なう GUI が構築できる。

### 3.2.5 動的可変性の実現

コミュニケーションの記述のところでプロトコルを協調させることについて述べたが、同様にして DeLis は内外部からその形態を変更することができる。

DeLis は、情報ユニットが発生する情報や他の DeLis や外界とのコミュニケーションで得られた情報に記述されたプログラムを実行することで、今まであった関数を変更することも、また新たに関数を定義することもできる。すなわち、その時の状態や要求に合わせ情報ユニットの生成変更が容易に行なえるということである。

## 3.3 DeLis による分散環境の構成

以上のように、分散環境を DeLis によって記述するわけであるが、その場合、分散環境は図4のように表すことができる。

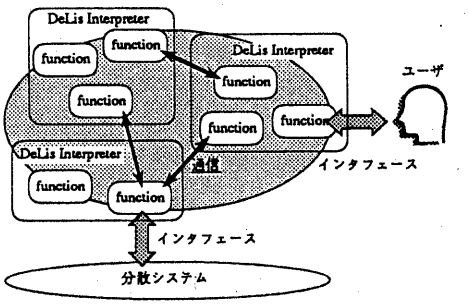


図 4: DeLis による分散環境の構築

# 4 評価

## 4.1 アプリケーションの記述

DeLis が、実際に分散環境の記述運営に適しているかどうかを調べるため次のような分散環境のためのアプリケーションを記述することによりその評価を行なった。

### 4.1.1 DoReMi

DoReMi とは、ネットワーク上の各種サービスを都市メタファーを用いて統一的に提供する異機種分散系上のグラフィカルユーザインタフェースのことである。[5]

DeLis を用いての DoReMi の実現を図 5に示す。

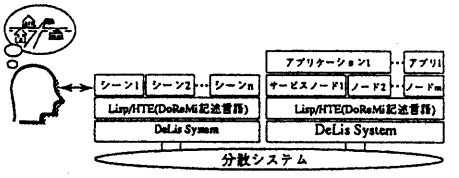


図 5: DoReMi の記述

DeLis によって DoReMi 記述言語 Lisp/HTE を記述し、その上で都市メタファーのシーンや、サービスノードの記述を行っている。

この DoReMi は C 言語によっても作成したが、この DeLis を用いることによって C 版の約 5 分の 1 の記述量とすることができた。また、C 版では不可能であった動的なシーンの生成といったものが可能となっている。

### 4.1.2 メッセンジャを用いたコミュニケーションツール

メッセンジャとは、自律分権協調概念に基づく計算モデル Kemari[9] においてメッセージ自身に自律性を持たせたものである。メッセンジャは、メッセージを受けとりそこに含まれるスクリプトを解釈し、それにしたがって分散環境上を自律的に行動する。これを用いてコミュニケーションツールを作成した。[6]

メッセンジャ自身を DeLis により記述し、さらにメッセンジャにわたすメッセージも DeLis の関数とすることにより実現されている。

ここで作成されたコミュニケーションツールは多数決システムであるが、この時のメッセンジャの動きを図 6に示す。

メッセンジャはその議題に関係する人物を探しながら自律的に多数決をとってまわるというものである。この動作はメッセンジャには定義されていない。メッセンジャに対して、メッセージとして多数決をとるようなスクリプトを渡してやることにより実現されている。さらに、その議題に関係のあるユーザを自律的に探して行

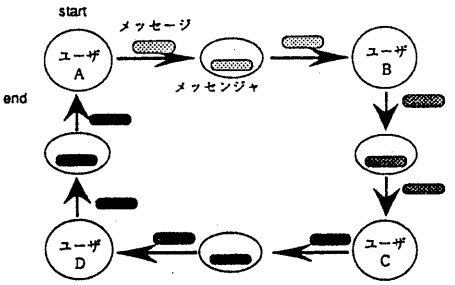


図 6: メッセンジャの動き

動する。すなわちメッセンジャの内部状態は常に変化しているのである。

### 4.1.3 分散リダクションシステム

ここでの分散リダクションシステムとは、メッセージ交換方式を用いたリデューサで LAN 上でリダクションを行なうものである。[7]

ユーザはリデューサに対してリダクションをさせる項の他に種々のメッセージを送ることができ、そのメッセージによりリダクション実行の制御を行なうことができる。すなわちプロセスの機能は動的に決定され、ユーザが積極的に実行改善のための制御を行なうことができるわけである。

### 4.1.4 InterCam

InterCam とは、Unix メールシステムを利用した、メール自身がユーザとのインタラクションを行なうコミュニケーションツールの一種である。[8]

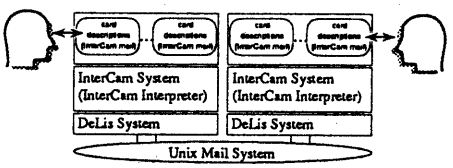


図 7: InterCam の記述

これは図 7のように DeLis 上に InterCam インタプリタである InterCam システムを構築し、

その上でUnix Mail Systemによるメールの中に記述されたcard descriptionを実行することにより実現されている。

## 5 考察、課題

以上のように分散環境というものが必要であることがわかり、その構築のための言語DeLisを作ったわけであるが、このDeLisによって容易に且つ効率良く分散環境のためのアプリケーションを記述できることが確認された。また、動的可変という柔軟性を持った運営を可能とすることができた。

しかし、その動的可変性によりリソースの保護についての問題も浮かび上がった。今まで存在し使用されていた関数に対して、外部から別の定義をされてしまうことがある。これによりそれまで使用されていた環境が正常に動作しなくなる可能性がある。これを回避するためには、ローカル変数のようにローカル関数的なものを用意するなどの必要性がある。

また、同様に関数の定義に関する事であるが、コミュニケーションを行なっている2つのDeLisにおいて片方に定義されていない関数が存在した場合は新たに定義することで解決できるが、双方のDeLisが同じ名前で別の定義がされているものを持っていた場合に、その識別をどうするかという問題がある。

今回作成したDeLisは試作版であり、細かな仕様はまだ未決定である部分が多い。先に挙げたようなアプリケーションを実際に記述することで、そのような細かな部分についてもわかりつつある。今後これに基づき、そうした部分の設計を行ないDeLisを改良していきたい。そのためにもさらに多くのアプリケーションを記述してみる必要がある。

また、パソコンなどの多機種、多種メディアに対応できるようにDeLisの拡張をおこない、これにより、例えばパソコンを含む異機種分散系上のためのUIの記述や分散ハイパーテキストの記述などへの応用を考えている。

## 謝辞

本システムを構築する上で共にプロジェクトに参加してくれている東北大学法学部3年金谷吉成君に感謝します。

## 参考文献

[1] Robert E. Filman and Daniel P. Friedman. *COORDINATED COMPUTING Tools and Technoques for Dstributed Software.* McGraw-Hill, Inc., 1984. (雨宮真人 尾内理紀夫 高橋直久 共訳: 『協調型計算システム分散型ソフトウェアの技法と道具立て』, マグロウヒルブック株式会社 (1986)).

[2] John Galletly. *OCCAM2.* Pitman Publishing, 1990. (丸山公雄 / 他訳: 『並列処理言語OCCAM2』, 日刊工業新聞社 (1991)).

[3] C. A. R. Hoare. *Communication Sequential Processes.* Prentice-Hall International (UK) Ltd., 1985. (吉田信博訳: 『ホーア CSP モデルの理論』, 丸善株式会社 (1992)).

[4] James W. Stamos and David K. Gifford. Remote Evaluation. *ACM Transactions on Programmming Language and Systems*, Vol. 12, No. 4, pp. 537–565, 1990.

[5] 伊藤眞. メタファネットワークの構築に関する研究, 1991. 東北大学大学院工学研究科修士論文.

[6] 五十嵐敏明, 布川博士, 野口正一. 自律分散協調型計算モデルを用いた コミュニケーションツールの記述. 情報処理学会研究報告, Vol. 92, No. 91, pp. 165–172, 1992.

[7] 寺島貴之, 布川博士, 野口正一. 分散lisp系を用いたメッセージ交換リダクション. 情報処理学会第45回全国大会講演論文集, pp. 5.7–5.8, 1992.

[8] 布川博士, 奥村成吾, 瀬川典久, 宮崎正俊, 野口正一. 分散lisp系を用いた電子メイルの記述. 電気関係学会東北支部連合大会講演論文集, p. 91, 1992.

[9] 矢野博之, 武宮博, 布川博士, 野口正一. 自律分散協調概念に基づく計算モデル kemari. 情報処理学会第42回全国大会講演論文集, pp. 5.75–5.76, 1991.